TÉCNICO LISBOA

Implementation of an Enterprise Architecture in SPMS using Enterprise Architecture Management Tools and Cartography Methods

Carlos Rodrigues Martins da Silva

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Pedro Manuel Moreira Vaz Antunes de Sousa

Examination Committee

Chairperson: António Manuel Ferreira Rito da Silva
Supervisor: Pedro Manuel Moreira Vaz Antunes de Sousa
Member of the Committee: André Ferreira Ferrão Couto e Vasconcelos

June 2019

I would like to dedicate this thesis to my mother Filipa Correia Rodrigues Martins da Silva and to my father, José Carlos Lopes Martins da Silva

Acknowledgments

I would like to thank my parents for their emotional and finantial support throughout my years at the great institution that is Técnico.
I would also like to thank both my siblings for their support and love during all these years.
I want to give a very special thank you to my thesis mentor, Prof. Pedro Manuel Moreira Vaz Antunes de Sousa, for the amazing opportunity to work in this specific project, and for the knowledge and guidance offered me during my time working on this thesis.

Resumo

Arquitectura empresarial é uma disciplina que se foca na estrutura e organização da informação nas diferentes camadas, tais como estratégia, negócio e infraestrutura IT de uma organização, ajudando esta a manter uma visão coerente dentro da mesma. No entanto, as organizações modernas demonstram dificuldades na obtenção de sucesso na implementação desta disciplina. Neste projecto, vamos abordar o caso específico de uma organização responsável pelo desenvolvimento e gestão de sistemas de IT relacionados com serviços de saúde e todos as organizações que utilizam os seus sistemas. Seguindo um paradigma de Cartografia Empresarial, e com recurso a um sistema de gestão de arquitectura empresarial, este projecto incluí uma análise às dificuldades relacionadas com a não existência de uma AE, específicas destas organizações, tais como problemas de comunicação de conceitos, gestão de diferentes fontes de informação e entidades, utilidade efêmera dos mapas organizacionais criados e a pressão política associada a projectos de mudança. Foram desenhadas soluções para os problemas identificados, sendo o foco destas a cobertura do maior número possível de problemas. Utilizando uma framework de CE, e a ajuda de um equipa de desenvolvimento, um conjunto de soluções, incluíndo novos métodos de registo de informação mais agéis baseados em formulários, batches para centralização em massa de informação proveniente de várias fontes, e um meta-modelo organizacional baseado em feedback, que criaram fundamentos sólidos para esforços arquitecturais futuros.

Palavras-chave: Arquitectura Empresarial, Cartografia Empresarial, Projecto, Base de Conhecimento, Sistema de Gestão de Arquitectura Empresarial.

Abstract

Enterprise Architecture is a discipline that focuses on the structure and organisation of information and the layers, such as strategy, business and IT infrastructure, within an enterprise, helping it maintain a coherent vision throughout all its levels. However, many organisations still struggle with succeeding with the implementation process of this discipline. In this project, we will focus on the specific case of the implementation of an EA project at a national healthcare IT system and management provider and the organisations that use its systems, all with no previous experience on the field of EA. Following an Enterprise Cartography approach, supported by an enterprise architecture management system (Atlas), this project includes an analysis to pinpoint the main struggles of the subject organisation, such as the non-existence of an architectural meta-model, difficulty in managing different information sources and entities and the effort of gathering said information, the short-lived value of views produced at a certain point in time, the negative perception regarding the use of Enterprise Architecture and political pressures regarding change projects. The later part of this project focused on designing solutions that would tackle these problems with a focus on the breadth of the solutions. Using an EC project framework and the help of an application development team, a set of solutions, such as easier ways to register information, batches for mass centralisation of different sources of information and a feedback based organizational meta-model, allowed for the creation of a solid baseline for existing and future EA efforts.

Keywords: Enterprise Architecture, Enterprise Cartography, Project, Knowledge-Base, Enterprise Architecture Management System, Meta-model.

Contents

List of Tables

List of Figures

Nomenclature

EA Enterprise Architecture

EA Team Team composed by the EA team on the organization side and the author.

EC Enterprise Cartography

eSIS Information Ecosystem

GDPR General Data Protection Regulation

KB Knowledge-Base

SPMS Serviços Partilhados do Minitério da Saúde

Chapter 1

Introduction

In this chapter, we will present the subject of this thesis, describing in detail the topic being studied, the motivation behind it's study, the specific objectives of this work and the structure of the whole document.

1.1 Motivation

The subject of this thesis is the implementation of an Enterprise Architecture (EA) project in a low-maturity organization [1] (an organization without defined architectural processes and/or these are poorly defined [2, 3]), along with entities that make up the information ecosystem eSIS (Ecossistema da Informação da Saúde) [4], that includes all public health entities that use the systems managed and produced by SPMS.

Serviços Partilhados do Ministério da Saúde (SPMS) is an organization whose business model is to develop, build and manage information systems and the information used by these to serve the Portuguese national healthcare service, which leads it to have a high production rate due to the constant demand by healthcare professionals and users. The project at hand is not only focused on the SPMS organization itself, but the whole network of healthcare entities to which SPMS provides systems to. This network is named "eSIS", which is an information ecosystem composed by all the public health institutions that adhered to this initiative (66+) and use systems produced by SPMS.

These systems are responsible for the management and production of large quantities of information, requiring a coordination between a nationwide grid of systems that share large amounts of information every day. Given the number of systems and quantity of data produced, altered and disposed of (as well as the care that some information requires), keeping track of all these variables becomes a difficult task. This results in a situation that the organizations from eSIS themselves are aware: the need to have a mechanism/platform that allows for an easier control and management of said information, and he cataloguing of that same information. EA is a discipline that provides a possible solution for this issue. Without an EA, SPMS recognizes that problems like the difficulty to communicate ideas, poor information quality, mis-coordination and lack of documentation exist. Given this, EA represents, for this organization, an opportunity to obtain more value and improve their functioning and that of the

ecosystem entities.

The theory behind EA encapsulates a set of agreed-upon frameworks, knowledge, information and methodologies that, if used correctly, signify a great gain of value for any organization willing to invest in it. EA benefits are well documented [5–8], with propositions such as the better alignment between different parts of an organization, improvement of business processes efficiency, more efficient and easier to perform communication, among others. The value proposition of EA seems to perfectly fit the organization's demand for a way to organize and standardize information throughout it. However, despite the promising ideal behind EA, the fact is that Enterprise Architecture projects have a very high failure rate (up to two-thirds of initiatives fail [9, 10]). Many authors theorize about the possible reason for such situation to be a reality, with organizations like Gartner [11] weighting in on listing the possible factors behind it.

SPMS accepted to participate in an EA project in order to implement an Architecture within the organization and the eSIS ecosystem, with the objective of consolidating all the information present in these entities, along with SPMS's, into a single shared platform, with the author being responsible for accompanying, providing input, managing (to a degree) and advising in this same project, all with the supervision of the mentoring Professor. The main issue is that since none of the organizations have any previous experience with EA or EA projects, many of the concepts and procedures behind such effort are not familiar to the them, which leads to appearance of many problems and difficulties. These issues are the focus of this thesis, as these will be studied and possible solutions for them will be proposed.

1.2 Topic Overview

SPMS and eSIS currently struggle with the absence of an EA. A large number of the systems and information SPMS produces and is responsible for do not have a proper documentation detailing their internal structure, nor the relations with other systems (this last one exists, but is too simple, an serves only to expose that, for example, system A "talks" with system B; details of the communication, such as what files are shared, are not present). This lack of detailed documentation about systemsand information proves very inefficient, since, for example, only members of a team responsible for a system have a vague idea of how it is structured. Communicating the structure to other teams, since no standard of communication or models exist, becomes a difficult task, not only because the other team may not understand the explanation or may get a wrong idea, but also the team itself does not have a single, absolute or clearly defined view of the structure of it's own system.

Aside from the structuring problems, problems with a lot of the information that is transferred between these system exist. This data also suffers from not being properly documented or registered, a situation that becomes a problem since it leads to cases where information is duplicated, systems that retrieve the same information from two different sources (faulty system design), and the overall disorganization in terms of who keeps what and who they take it from (regarding information). This particular problem became clear when the need for organizing and classifying information for the General Data Protection Regulation (GDPR) initiative imposed by the European Union arose, and the organization had no way

to provide all the information necessary for auditing operations.

Alignment problems were also noticeable, as most of the teams had a hard time explaining what business processes and products their systems where supposed to support, and how those which perform these processes use their systems.

Members of the ecosystem entities are, due to not being specifically concerned with the IT infrastructure and overall architecture, even less aware of these issues.

Given this context, EA represents then, for these organizations, an opportunity to obtain more value and improve their functioning. The theory behind this discipline encapsulates a set of agreed-upon frameworks, knowledge, information and methodologies that, if used properly, signify a great gain of value for any organization willing to invest in it. However, despite the promising ideal behind EA, the fact is that Enterprise Architecture projects have a very high failure rate [9, 12, 13], with up to two thirds of EA initiatives resulting in failure [10]. Many authors theorize about the possible reason for such situation to be a reality, with organizations like Gartner [11] weighting in on listing the possible factors behind it.

This specific organization finds itself struggling with the implementation of this project, since it represents a whole new reality for the it and the concept of EA had never been a part of it's culture. The problems identified throughout the ongoing project by the author result from direct field observation, as well as the analysis of common problems that occur in this type of projects (lifted from research and then compared to the reality of SPMS).

The following subsections provide a description of the identified problems this specific EA initiative suffers from.

1.2.1 SPMS and Ecosystem members show lack of motivation to perform the EA effort.

This problem could also be phrased as the difficulty to see value in Enterprise Architecture. Despite the promises of helping the organization align all it's components, map their reality and help plan future decisions, the individuals that are part of this organization and ecosystem seem reluctant of the benefits that EA can bring to the table. Not in the sense that the benefits are not seen as good, but that seeing that most EA projects are long-term efforts that take time to produce perceptual results (and sometimes, the benefits are not ”palpable”), some of the participants lose their interest in the efforts, seeing them only as bureaucracy.

Many times, when presenting and ”selling” the project to the members of the organization, the author was faced with questions like ”Yes, this seems interesting, but how does it concretely help us?”, and upon further discussion, while the members admittedly found some aspects interesting, the idea of having to swift some time and effort from daily tasks to this side-project (in the perspective of those belonging to the organization) seemed more of a bureaucratic novelty than an actual transformative action. This seemed to signal the need for a more persuasive approach to towards those less convinced of the benefits of the project. One possible explanation for this issue is that EA is a very C-level (Chief) and M-level (Management) oriented discipline, meaning that those who are the end receivers of the information are

typically people with decision-making responsibilities, while at the same time, the information retrieval is performed by the operational workers, who feel that the effort of registering and collecting information does not directly benefit them (at an operational level). This can cause frustration that possibly leads to the lack of motivation to perform EA or other activity that helps improve the it's state.

The question then, is how can we prevent this situation from happening. What tools or methods can one implement in an organization in order to convince these individuals that EA is a very important subject and that a correct information collecting and sharing (done via EA maps/views or other features) can help not only improve the overall state of an organization, but the operational worker's function as well.

1.2.2 Information collecting for EA reveals to be a cumbersome task

In order to successfully implement an Enterprise Architecture, one should first have a "Knowledge base" [14](concept explored in Section 3) or centralized information source, that may or may not feed from other sources. This base represents the whole information that makes up the EA, the source from where EA collects all the information it needs to describe the organization, it's future plans, dependencies between concepts and instances, among other things. The problem is that this organization does not have a fully standardized way of storing information, or in some cases, the information itself does not exist. As verified, in most cases, when management or some other department required a list of information regarding some system or category of information, a new information collecting effort would take place. Not only was the effort of collecting information needed again, but that same process also proved to be very inefficient, with large quantities of Excel sheets being manually completed, previous sheets had to be evaluated to check their current validity (to check if the information present still relevant/accurate), different sources had to be checked, ranging from Microsoft Word documents, PDF files, printed models that had once been created, etc.

This obviously translates into a very unproductive situation. This situation implies the need for an effort that is equally proportional to the complexity and amount of information this specific organization works with, which is yet to be properly evaluated. In an organization that has a large number of systems, dependencies between these systems and large quantities of information in said systems,this can represent a huge effort just to collect the necessary information to, not only map the EA, but to perform tasks such as decision making.

Modern EA efforts rely on EA Tools in order to represent, and in some cases, store this information, however, depending of the capacities on said tools to incorporate sources of information to populate the knowledge base, the effort to collect information can be easy or more difficult. In both cases, the way an organization's information is organized is also a factor, and in the worst case, the organization has no way of connecting the sources of information to the tool, which for the benefit of this project, is not entirely the case, meaning all information will not have to be inputted manually. This, however, will only slow down the whole process, since it is inevitable, as the tool being used cannot work without a knowledge base.

The question then, is how can we use the tool or other resources to speed up the process of information collecting, while at the same time making it a more ”pleasant” endeavor.

1.2.3 SPMS struggles to find a way to use EA to solve operational problems

Closely related to the problem mentioned in the subsection 1.2.1, in the sense that operational workers do not see this project as something that will directly benefit them, since the information produced by EA is more oriented to upper hierarchy levels of an organization, and given that EA is a governance oriented discipline [15], it's main focus is of course, not (directly) the operational part of and organization. This situation creates a sort of ”conflict” of interests between the upper hierarchical levels and the lower levels.

The organization struggles to understand and/or find a way to use EA resources, such as maps and metadata analysis to directly help perform efforts of an operational level. Finding this ”missing link” could translate not only in a more efficient use of the existing resources, but also provide more support to those operational tasks.

Also, given that a large number of standards, frameworks and regulations this organizations must comply with, the use of EA to respond to these requirements seems like a natural fit. The challenge lays also on creating ways of using EA to overcome these obligations.

1.2.4 Views generated by EA become obsolete very fast

In this day and age, organizations change at a very fast rate [16], and SPMS, due to the nature of it's business, is no exception, meaning that the creation of a static map based on the information collected about an organization may become obsolete in a short-term period. New projects are approved, structural changes occur, new departments are created etc. This all leads to an ever changing landscape that becomes hard to represent, given how ephemeral the representations of the present are.

As of June of 2018, SPMS had around 16 structure altering ongoing projects (in different progress stages), most of which have no representation in the existing maps, being that the only source of information about these are the teams responsible for their development, as well as ongoing documentation that is created. The main issue with this is that this same documentation is more focused for internal (team) use, and not for the overall organization (terms and concepts used are defined inside the team, and are not shared), meaning that if the team wishes to connect to another existing system, there is no information source, in the form of an architectural view, that can provide information about which components should be used for that connection, or the other teams have no way of knowing how the new projects will affect them, unless extensive technical meetings take place. And even when the teams manage to agree on a set of terms and concepts, maps built, due to their static nature, have a short lifespan, since requirements and specifications of the projects change during development, and constantly creating new maps becomes more of a chore than an actual benefit for the projects' development.

This problem is intensified by high production and change rate characteristic of SPMS (new demands and requirements translate into changes to the systems and/or creation of new ones), since

these changes require a change in the structure of the information systems of SPMS, which in turn make the previously created static maps obsolete.

With these problems, other one rises: even if we manage to bridge the representation of the present with the ones in the future, meaning that even if a way of mapping present to future stages is created, this bridge should take into account the volatility of change. Some unexpected event/decision can happen that drastically changes the previously built connection between present and future.

To sum up this issue, the main roadblock to the use of maps in an organization like SPMS, is the almost constant need to build need maps or add elements to existing ones. Finding a way

1.2.5 No uniform language to define concepts of the organizations reality

One of the main issues the organization faces is the lack of a unique way of communicating information, not in terms of tools or ways to transmit it, but how this information is conceptualized, since there is no defined structure for the information i.e what does the concept of ”system” represent, what does that concept include, how does it relate to other concepts, what influence can stakeholders have on it, which leads to more questions, such as what type of stakeholders exist, are they all the same or are there categories to define them?

This lack of structure causes communication problems, since, for example, two members from different teams, when discussing some concept, such as internal structure of a system, may not have the same mental model of each part of the system and what it means in the bigger picture. Member from team A can have a hierarchical view of the system, where each level represents greater detail in functionality or importance, while member of team B sees the system as a pool of small boxes, where every box represents a functionality of the system. This represents a situation where two different people have different views on how thing relate to one another. This can be a problematic situation when trying to integrate different components of two different systems, not only because the wrong information can get across as the right one, but the need to ”translate” both ideas only slows down the whole process. The aforementioned situation is considerably common within SPMS, as revealed by the meetings with different teams.

When trying to find consensus on the concepts of the meta-model and what they should represent, opinions varied with great frequency. Using the system example again, for some members, a system represented the whole SPMS information structure, seeing SPMS as a giant system, that aggregated different ”solutions” (such as e-Prescription, National Patient Registry (RNU) etc.), while at the same time, elements like service buses where considered isolated systems by some of the members. This disconnect between views imposed a roadblock in communications. Another dimension of the problem, besides the lack of agreement, is a more political nature, since for many of the workers, their perspective of things such as systems has been used by them for a long time and all the previously developed work was done using this mindset. Even when agreements between teams occurred in meetings, the terms where later ”translated” to the team’s or individual perspective. Changing this perspective also proves a problem, since many of the workers proved themselves to be resistant to change in this aspect.

What concepts are important to map, what are the dependencies between said concepts, what rules they abide to ? In an ideal EA setting, every member of the organization and ecosystem would agree on the importance and some concepts and the way they should be represented. This, however, is utopian. In the worst case, every single member as a different idea of what is important and what is not, however for this specific case, many people tend to converge to certain opinions about concepts, specially within their department/group of work, with some concepts are already cemented (Ex: most teams agree on what a system is).

The challenge of creating a meta-model for an entire ecosystem is to align all the visions that exist within it. This consensus is somewhat troubling to find, specially in organizations that have no standardized collected data. Different groups of people have different ways of conceptualizing their systems and concepts they work with, which in turn results in some level of disconnection between views. Of course, not every single member of an organization needs to have a complex grasp on what the meta-model is and what each single concept represents, but even if we restrict this task to at least one member of each group/department, there still exists a gap in consensus. Creating a democratically accepted (with a large majority) meta-model proves to be a challenge. Another part of this challenge is deciding if only ”specialists” (those who are more familiar with EA) should define the meta-model, if everyone’s opinion should count, or if a balance is between them is needed and to what degree should each opinion be valued.

1.2.6 Lack of awareness of what the main point regarding the management of an EA effort should be

Since SPMS has no previous experience with EA projects, knowing where to correctly focus efforts and resources can be a challenging task. Should information collecting be the main focus? Is the meta-model being built a definitive one or should it offer some flexibility? Pinning down certain areas where most of the members of the organization have more difficulties can also be a demanding task in the initial phases.

Given the failure rate (about two thirds [10]) among EA projects, the commitment to an effort of this nature requires a set of preoccupations that organizations must have, main worries and points those responsible should be specially aware of. However, those are not formally defined within the context of EA projects, with many being theorized but never effectively confirmed [17].Given the specific context of this project, what should be the main preoccupations and concerns an organization, and more specifically the project manager should have? What methods can be used to make sure that these problems are formally documented, so that can can be more easily mitigated in the future, assuring the organization is prepared to tackle them.

1.2.7 Difficulty coordinating multiple repositories

Perhaps the the most challenging problem this project is facing, is the coordination of the information coming from various sources and to be imported from and into multiple repositories. The goal, like

stated in sections 1.1, is to consolidate the whole information of the 66+ repositories, including SPMS's individual one, into a single repository, adequately named eSIS, following a federalized approach. Given that face-to-face meetings with every single health entity to present the whole project, as well as the theory behind it, specific cases and other concerns with the information is impossible, the information produced and imported by some of the entities will inevitably be bound to error. Some of the entities have already been in contact directly with the project team, while others are planned, most of them will only be given official documentation, due to time and budget constraints. This of course, proves inefficient, but for now necessary.

The issue is then finding mechanisms, features and methods that can allow for the diminishing of problematic situations, and of course, being able to successfully merge the information in each individual repository into a single one, as well as monitoring the information imported to each repository to be certain that it follows the established rules and principles.

1.3 Objectives

The objective of this thesis is to successfully identify and correct problems that occur during the development of the project at hand, as well as solving problems the organization faces due to the lack of an existing EA. As mentioned in the previous subsection, problems have surfaced while working with the project, some more urgent than others. Part of the research is to identify this urgency, study these issues, understand why they exist, possible causes for them, and how to, with the available resources, mitigate or completely eliminate them.

This thesis also focuses on, for this specific project, evaluating the effectiveness of the solutions, and study the affect these have on the overall project. It is important to mention that, disregarding the implemented solutions and the outcome of these, this thesis' scope is this specific project. While some conclusions may or may not be true for other EA projects, all problems and solutions are studied in an "isolated" environment/context and should not be taken as generalist conclusions, since that would require empirical validation on a plethora of cases.

For this thesis, due to the nature of the project and the previously agreed terms of the contract that define this effort, Link Consulting's tool Atlas(Fig 1.1) will be used as the primary EA Tool. Atlas is a Enterprise Architecture System(EAMS) that allows users to define their meta-model, by creating classes and relations between classes, as well as allowing for the analysis of meta-data, architectural scenario creation and architectural maps visualization. Atlas also allows for the input of information from various sources (such XML and Excel files) into a single repository where all is stored.

1.4 Thesis Outline

The thesis will will abide by the following structure:

1. Introduction: Thesis themes are presented, the problems identified with the subject organization

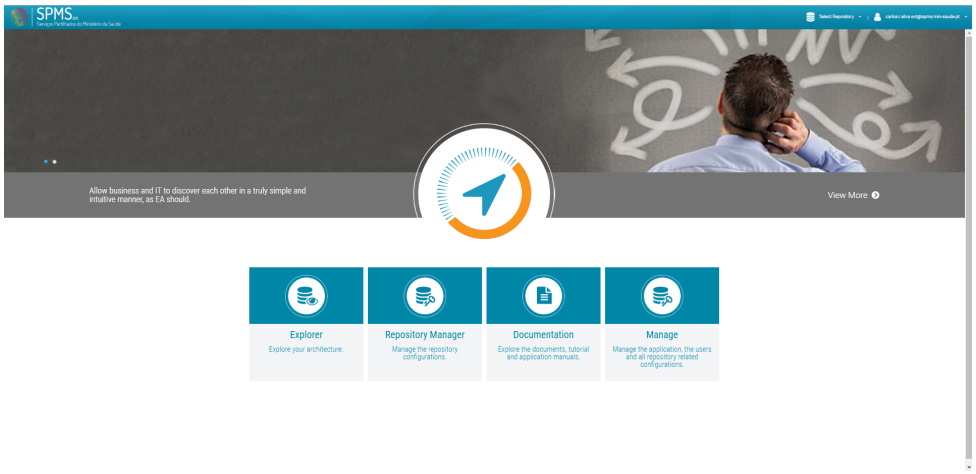

Figure 1.1: Atlas tool's main page

are described in detail.

2. Background: Relevant literature will be presented. Topics explored in this section will be of use for the research performed throughout the project, and served as the basis for the solutions presented in the Implementation chapter.

3. Implementation: In this chapter the solutions proposed and implemented during the project will be explained in great detail, such as the motivation behind it and how it was implemented.

4. Results: The results of the implementation will be exposed. Insights about the results and exploration of these will be provided to the reader.

5. Conclusion: Final words and thoughts on the whole of the thesis will be summarized.

Chapter 2

Background

In this chapter, the sources considered essential to the development of the theme of the thesis will be explored, providing insight on how the information within these publications can help us better understand the issues we are trying to tackle. Each section of this chapter refers to a particular research topic, and the most relevant publications regarding that topic will be presented, along with their importance to the project.

2.1 Enterprise Architecture

According to **Federation of EA Professional Organizations** [18] Enterprise Architecture can be defined as ”a well-defined practice for conducting enterprise analysis, design, planning, and implementation, using a holistic approach at all times, for the successful development and execution of strategy. Enterprise Architecture applies architecture principles and practices to guide organizations through the business, information, process, and technology changes necessary to execute their strategies”. Other definitions exist, for example **Lankhorst’s**: ”A coherent whole of principles, methods, and models that are used in the design and realization of an enterprise’s organizational structure, business processes, information systems, and infrastructure” [19];

Enterprise architecture focuses on finding the optimal structure for an organization’s IT, with the discipline not being limited to that particular field. Lankhorst states that EA goes beyond the ”local” perspective of organizational problems, since ”Enterprise architecture captures the essentials of the business, IT and its evolution. The idea is that the essentials are much more stable than the specific solutions that are found for the problems currently at hand. Architecture is therefore helpful in guarding the essentials of the business, while still allowing for maximal flexibility and adaptability. Without good architecture, it is difficult to achieve business success.” [19]

EA therefore provides economical value to an organization. According to Erwin’s Zac Cole [20] the value of Enterprise Architecture fundamentally rests on it’s ability to (when properly implemented) align business and it’s operations with IT, making the practice of EA the bridge between the organization’s current and future states. This alignment is also valuable in understanding how an enterprise can shape

it's internal and external forces, as well as connecting the IT component with the business practice. This, along with frameworks such as the Strategic Alignment Model[21], can help an enterprise clearly define it's strategy . The author also points out four core indicators of the practice's economic value to an organization:

Improved Strategic Planning, since EA allows for an organization to align a plan with it's implementation;

Bettering Communication by allowing every (relevant) actor to see and/or share information regarding their own micro-context within an organization. The tool used to model and manage the Enterprise Architecture efforts comes into play in this specific indicator, since the capacity to communicate ideas and information is highly dependent of the tool's capacity to show and manage information; Other authors [22, 23] also mention the importance of the existence of a common ground in terms of terms and concepts within an organization.

Tactical Improvements, by exposing the structure of information and processes in an organization, one can verify what is aligned and what is not, allowing for more accurate improvements;

Taking Better Risks. More and better information equals more informed decisions, If an organization has a clear idea of their strategy and how it is structure, decisions on it's future or directions to take become an easier task, with the possibility of designing possible scenarios of what the future should look like. Other authors like Marijn Janssen have also found that EA plays a big role in risk management, saying "EA can be viewed as one type of risk management instrument that is complimentary to risk mitigation in project management approaches" [24]

A Gartner publication [25] shows numbers that reflect the kind of investment companies are doing in the field of Enterprise Architecture: as of the date of the referenced publication, 70% of the organizations are either starting, restarting or renewing their EA related efforts, showing that this value is noticed by organizations.

2.2 State of the Art

In [23], the author provides a deep insight on the current trends (as of 2013) in EA and explores how the discipline is positioned within the context of corporate and IT governance.

In terms of methods and frameworks, the authors present the most relevant ones. Frameworks provide valuable information on what should be part of an EA, however, as the authors states:" to ensure the quality of the enterprise architecture during its life cycle the adoption of a certain framework is not sufficient". This points to a more flexible approach towards EA, as it is not a field that benefits fundamentalist applications, i.e, always using the same approach will not produce the same results. The authors mentions several relevant architecture methods/frameworks, such as **Enterprise Unified Process (RUP)** [26], which the main point is that it offers a iterative approach to architecture development, instead of the traditional "waterfall"; **UN/CEFACT Modeling Technology** [27], which focuses solely in business processes, utilizing the **Business Collaboration Framework** [28]; TOGAF [29]; Zachman Framework [30] and IEEE 1471-2000 [31].

The Zachman Framework offers a way to structure an enterprise through the use of a ”schema” that the author refers allows to map” the intersection between the roles in the design process: that is, owner, designer, and builder; and the product abstractions: that is, what (material) it is made of, how (process) it works and where (geometry) the components are relative to one another”. This framework, however, is not the a methodology, since no information is given on how to collect or manage the information presented in it. The author refers that the advantage of this framework is that it is easy to understand, while at the same time describing the enterprise in holistically, without being constrained by specific tools or methods. However, as the author states, the large number of cells resulting from this mapping can become overwhelmingly complex, denying the initial benefit of being easy to interpret, and the relations between these are not clear.

The IEEE 1471-2000 is a formulated standard that describes what rules an Architecture for ”software-intensive systems” should follow, offering ”a solid theoretical base for the definition, analysis, and description of system architectures”. The main offering of this standard is the conceptual model of architecture description. This model offers a list of concepts integral to the description of an architecture, and how these concepts influence each other:

This conceptual model allows to understand the dependencies that exist within an architecture, and how the decision made in the context of one of the concepts can affect and be affect by others. Presenting a holistic view of concepts allows also for a better perception of the whole that constitutes an architecture

This standard also offers six architecture description techniques:

1. Architectural documentation referring to identification, version, and overview information.
2. Identification of the system stakeholders and of their concerns, established to be relevant to the architecture.
3. Selection of architectural viewpoints, containing the specification of each viewpoint that has been selected to organize the representation of the architecture and the reasons for which it was selected.
4. Architectural views corresponding to the selected viewpoints
5. Consistency among architectural views.
6. Architectural rationale for the selection of the current architecture from a number of considered alternatives.

These description techniques seem to offer broad enough recommendations to guarantee a collection of information that respects all the stakeholders, maintains consistent information independently of the chosen viewpoint, and how the latter serves the first.

TOGAF’s ADM is an iterative and generic (meaning it can be used on any enterprise, requiring only an adaptation of the concepts to the enterprise’s reality) method for the development of an enterprise architecture. This method can be used in conjunction with other methods or frameworks, namely the Zachman Framework. This method is composed of ten phases, all of them iterative and iterated on:

1. **Preliminary:** Determine and Establish Architecture Capability
2. **Architecture Vision:** Develop high-level aspirational Architecture and obtain approval for a Statement of Architecture Work.
3. **Business Architecture:** Develop the Target Business Architecture.
4. **Information Systems Architecture:** Develop the Target Information Systems Architectures.
5. **Technology Architecture:** Develop the Target Data Architecture
6. **Opportunities and Solutions:** Determine whether an incremental approach is required, and if so identify Transition Architectures that will deliver continuous business value.
7. **Migration Planning:** Finalize the Architecture Roadmap and the supporting Implementation and Migration Plan
8. **Implementation Governance:** Perform appropriate Architecture Governance functions for the solution.
9. **Architecture Change Management:** Ensure that the architecture lifecycle is maintained and that the Architecture Governance Framework is executed.
10. **Requirements Management:** Manage architecture requirements identified during any execution of the ADM cycle or a phase.

For each iteration of the ADM, it is necessary to review the following points: The breadth of coverage of the enterprise to be defined; The level of detail to be defined; The extent of the time horizon aimed at, including the number and extent of any intermediate time horizons; The architectural assets to be leveraged in the organization's Enterprise Continuum, including assets created in previous iterations of the ADM cycle within the enterprise and assets available elsewhere in the industry.

This iterative approach allows for constant improvement of the architecture, since it allows the implementation of changes resulting from the feedback of previously implemented decisions. This situation creates then a looping-cycle of self-improvement that helps an architecture grow.

The author also exposes some of the **description languages** used, such as BPMN, UML and ARIS, however, it is stated that none of them actually managed to become the standard for description of architecture, since according to the author, most of them suffer from problems such as the lack of a strong formal basis, poorly defined relations between domains and the overall architectural vision is not achieved, since they tend to focus on specific sub domains such as business, applications or technology.

Service-Oriented Architecture (SOA) is a paradigm that follows a "set of design principles that enable units of functionality to be provided and consumed as services. These services provide the 'units of business' that represent value propositions within a value chain or within business processes. This essentially simple concept can and should be used not just in software engineering, but also at all other levels of the enterprise architecture, to achieve ultimate flexibility in business and IT design". The author also goes on detail on why SOA provides such benefits to the practice of EA, stating that the concept

of service is something that is understood in the different domains that make up the enterprise, which facilitates communication. A SOA approach also enables a greater degree of interoperability, flexibility, cost effectiveness and innovation power, since everything is separated in such way that the degrees of dependency between components are so low that one can take a piece and put another without great consequence. Also, modeling such structure is easier than the traditional highly-interdependent systems, since everything is simpler: one artifact has self-contained atomic function, that when combined with other, creates a whole capable of delivering more value. If an architect wishes to map a complex system, he/she must simply describe it from the most atomic function to the most complex, using these atomic components together to map greater functionalities (almost analogue to building a Lego figure).

2.3 Enterprise Architecture Tools

The use of software tools to support EA is fundamental, since they allow for an easy and fast creation of architectural views. These tools, due to the demand of more complex features and deeper analytic capabilities, have evolved, but a low market offer exists, when compared to other types of software. Tools like Enterprise Architect [32] by Sparx Systems, IBM's System Architect [33] and Plainview's EA solutions [34] were studied in order to understand the state-of-the-art in terms of EA tools.

Enterprise Architecture tools have a very high degree of dependency on the information that is imported/inputted in them, making the human factor an influencer in the EA process, which leads to positive and negative contributions: the fact that the process is not fully automated, it is always subject to human error and mistake, such as incorrect and/or inconsistent information. The positive side of this issue is that human intervention allows for information context, i.e, a humans can have a certain perspective of the information and where it belongs, what format or standards it should follow, and more importantly, use that information in ways that help the organization's goals. As stated in the Gartner publication "Enterprise Architecture Tools Are Positioned to Deliver Business Value" [35]:"Enterprise architecture tools can provide tremendous business value, but only when aligned with the needs of the organization." The human factor is what decides the needs, since an organization is a collective of people trying to satisfy its own needs and the needs of those they serve.

Valuable features were also researched, with some authors offering new solutions for the improvement of tools. Authors in [36] focus on analysis capacities a EA tool can provide. Concepts like security, availability and reliability cannot be traditionally described in an architectural model, concepts that are relevant for an organization to know when describing their structure. The authors then propose a new methods that allows for the quantification of these concepts. The creation of a "abstract model", composed by entities, attributes, relation between entities and relations between attributes. This chain of dependencies is what the authors defend can help quantify (using an arbitrary scale) concepts like security:"As an example, the experience of the system administrator may affect the availability of the system he or she administrates. The system's availability can in turn affect the availability of the provided functions, which is the quality attribute of interest"

Authors in [19] provide an insight on the state of EA tools, affirming that while there was evolution,

there is little offer in the market. Interoperability is also cited as the one of the main challenges that tools still struggle with, since each tool is intended to be used within its self-defined model. From research of other tools, this seems to be the case, since there is not one tool that fully embraces all aspects of EA, with some focusing more on the visual aspect, others Quantitative analysis is also explored, with the author proposing a model based on the evaluation of the relations between classes. In one of the examples, the author wishes to calculate the time a certain business process takes. By attributing a time value to each of the concepts involved (Ex: actor performing access to application takes 5.0s, application contacting web-service takes 0.2s etc.) the author is capable of deducing how fast the process is. This offers a new depth to the analysis of view, since it becomes possible to extract more information than just dependencies between elements

The analysis of different tools and current features considered the norm allowed for an insight of what might still be needed and what already exists within the market. Features like the metadata analysis where absent from a lot of the tools (capacity for the tool to produce information based on what is present in the views). The ability to conjugate multiple frameworks or integrate them in the tool was also a ”common” flaw, since tools tend to specialize in a single aspect of EA and are meant to be used with others, while at the same time not offering an integration between these. This disconnect and lack of embracing EA holistically (as a discipline) becomes a problem, although the argument for the cost of developing such a tool is understandable.

2.4 Enterprise Cartography

Formalized in [14], Enterprise Cartography (EC) provides a new approach to traditional practice of EA. In this publication, the author exposes this new concept from the theory behind it to the actual implementation in real-life projects.

The authors present EC as a new concept that does not regard itself with the purposeful design normally associated with EA, focusing instead solely on the representation of an enterprise's reality. According to the authors ”with the introduction of the concept of EC, we can decouple the problem of design from the problem of building a consolidated set of representations of the enterprise. Different architects can use their preferred design approach, leaving up to the cartographer the task of consolidating each partial architecture into a single global set of maps representing the enterprise architecture of the AS-IS as well as of the emerging AS-IS.”

The concept of emerging AS-IS is explained along with other important concepts behind EC. According to the authors, principles behind EC are:

1. **The enterprise's metamodel**. This includes all the artifact types and allowed relations.

2. **Architectural statement**, a proposition stated using only the relations and artifacts described in the metamodel.

3. **Architectural Map**,a visual representation of architectural statements.

4. **Alive Artifact** refers to an artifact that as an active part in the organization, participates in architetcural statements
5. **Transformation Initiative** is set of planned activities that causes change in associated artifacts.
6. **Enterprise observation** is a construction of architectural statements through the observation of the enterprise's reality.
7. **AS-WAS** is the state that relates to enterprise realities at a given point in the past.
8. **Knowledge Base (KB)** repository that holds metamodel, architectural statements and the conceptual maps.
9. **AS-IS** state that describes enterprise reality in the present.
10. **emerging AS-IS** is the state of the enterprise after the completion of ongoing transformation initiatives.
11. **TO-BE** state of the enterprise at a certain point in the future.
12. **Enterprise Architect**, person responsible for the design of the enterprise; makes architectural statements about the future.
13. **Enterprise Cartographer**, person responsible for the collection of architectural statements from observations of the enterprise's reality.

The authors separate the concepts of cartographer and architect to further push the point that this new paradigm sees design and cartography as two separate entities. The introduction of transformation initiatives in the maps created by cartographers also offers a new perspective on the traditional way to separate past, present and future. According to the authors, the plans for the future must be part of the present. The benefits of this proposal become clear using the following analogy: If a construction company wishes to build a road from point A to point B, it can do so, and establish the plan and budget for it, and even start it's construction. But if, in the meantime, some other company builds an infrastructure in the path of the road, the construction company must recompile their plans and try to solve the situation, something that in the real-world could translate into a loss of money. So if the company wishes to successfully plan the road, it must also know what other projects are currently planned for the area, and ensure that preventive measures are made, and careful planning is thought out. By incorporating the plans of the future in the maps of the present, cartographers provide the enterprise with a risk, strategy and integration management tool that is useful for decision making and communicating the different projects between parts of an enterprise. This point is part of the author's EC fundamental principles:

1. **Transformation initiatives are enterprise artifacts**, since the authors intend to represent them, they can participate in architectural statements.
2. **Changes in the set of alive artifacts are planned ones**, meaning transitions between states are planned.

3. **All enterprise artifacts have a five-state life-cycle**: conceived, gestating, alive, dead, retired
4. **The TO-BE state precedes the AS-IS state**, since for artifacts to be alive, they must exist in the previous stages before.
5. **The emerging AS-IS can be inferred by observing the AS-IS of an enterprise**, through observation of transformation initiatives and other artifacts

The authors also offer a template for the realization of EC projects, based on the theory exposed and field experience. The proposed template is divided into six phases:

1. **Identify project goals**, define the scope of the project and understand what is trying to be modeled.
2. **Define the meta-model**, understand the most important artifacts and the relations between them
3. **Identify the best sources of information**. After defining the metamodel, find where information is keep (ex: structured repositories).
4. **Structure the processes and tools to capture information**. Setting up the methods to import information from the sources to the KB
5. **Define and Configure Architectural Maps**. Definition of the maps that allows views relevant to the client.
6. **Populate the KB with initial baseline**. Loading the information from sources using developed methods, in order to populate KB.

The influence of EC in the organization's governance is also explored by the same authors in [37], where the authors describe an organization as an organic whole, composed by people and computers, following [38], and model the functioning of an organization as a systematic feedback-loop between what is sensed (in this case the feedback of the actions) and what is inputed (actions in reaction to feedback).

According to the authors, "Enterprise cartography denotes the discipline dealing with the conception, production, dissemination and study of enterprise maps to support the collective understanding of a dynamically changing organization", meaning EC serves this feedback loop as a bridge between those o make decisions and the effect of their decisions.

2.5 Enterprise Architecture Projects

In [10], the author provides an insight on the causes for the very high failure rate of 66% found in EA projects. The author shows that according to the study at hand, in which 161 correspondents from 89 organizations (from various industries) were interviewed, around 92% believed that "EA should be determined by the vision, strategy and objectives of the company", while at the same time only 40%

actually included those same concepts in their respective organization's Enterprise Architecture project. This situation seems to point towards a case of misalignment between an organization's goals and strategy.

The author "blames" this situation on the following factors:

1. Lack of EA awareness
2. The fact that setting up an architecture took longer than expected.
3. Not enough support from C-level (CIO and CFO for example) so that EA is not given enough status and expectations cannot be fulfilled in practice.
4. Limited commitment from interested parties so that there is a return to old habits, and agreements are not complied with.
5. Not enough EA awareness among interested parties inside the organization. EA not a generally accepted concept in daily business activities.
6. Financial and political issues that thwart EA projects.
7. Setting up an architecture takes longer than expected. This means it takes longer for the results to become visible, which means there is a considerable risk factor for EA.

This conclusions show that, while organizations see the potential of EA, none seem to actually understand what is necessary to make it work. Aside from the political issues, the expectations towards EA seem misplaced, with organizations expecting results without the necessary commitment to make it relevant.

The theme of misplaced expectations and lack of understanding of what makes EA work is also explored by [17]. Presenting an insight on what are the main factors that make EA "work", the author points out that EA in nature, does not follow the typical engineering field, in the sense that there are no rigorous guidelines on how the activities should be performed, and that different projects with the same input do not produce the same results, while also providing a critical position towards the theory-heavy approach many of the literature as to EA, mentioning that many authors provide possible solutions, but no general definitions exist. The author then concludes saying that the main critical factor to make EA work is not a set of smaller factors, but instead: "How architecture is practiced, how it's routines are performed, is the Critical Success Factor for architecture". Pointing toward a more hands-on approach to EA, the author states the clear definition of architectural processes within a specific context (organization) is the truly defining factor for success.

In [39], authors also point out important mistakes to avoid when managing the construction of views that support and architectural endeavor, namely oversized maps since "oversized information models cannot be maintained in the future, as e.g. the maintaining group neither might be aware of the causes due to which certain concepts were introduced to the model nor might know what the concepts are used for", with the authors mentioning these should be completely avoided.

Authors in [9] present a proposal for the effective management of Enterprise Architecture projects, while also providing a valuable list of factors that distinguish EA projects from other types of projects:

1. **Project:** usually associated with organizational processes and systems.
2. **Structure:** Usually several projects depend on these with a high degree.
3. **Scope:** Less Defined and constantly under change
4. **Change Control:** Changes can be defined but are difficult to track.
5. **Stakeholders:** Large number and difficult to identify.
6. **Resources:** Efforts are ”part-time” (not the main preoccupation).
7. **Risk:** High Impact, but difficult to identify.
8. **Budget:** Poor
9. **Evaluation Indicators:** Subtle, more qualitative and difficult to measure.
10. **Speed Changes:** High

This list provides valuable information on the factors that influence projects expectations and concerns throughout their development. The authors formulate that EA projects are in nature difficult to structure:”we can conclude the inherent different nature of enterprise architecture projects with other projects is the fundamental factor of distinction and this type of project in addition to the technical nature, organizational nature and their management is complex and requires to creating dedicated project management framework for enterprise architecture projects”. As pointed out in the list, authors defend that EA projects do not follow traditional project management conventions.

Chapter 3

Implementation

In this chapter of the thesis, we will provide insight on the implementation of the methods and solutions formulated for the already identified problems, as well as other issues that arouse during the project development.

In order to better mitigate each of these problems and how each solution influenced these, we will dedicate a section to each of the solution points, fully exploring all the issue and concepts involved.

3.1 Standardization of Concepts/Language

The issue of being able to effectively communicate ideas throughout the whole organization was, as previously mentioned, one of the main issues this organization faced. In order to tackle this issue, the author proposed the need for the conception of a unified language, with the objective of it being able to accurately represent the organization's reality and to be used by all members of the organization when referring to real world concepts.

Since this is a EA project, the logical decision was to advance with the construction of a metamodel, which according to the EA project template presented by the authors of [14], is one of the initial tasks (second, after definition of project goals). The analysis, building and developing a set of rules to organize information and concepts regarding the context of SPMS was a task that had to be performed not only at the beginning of the project, so that a modelling basis for the rest of it could exist, but also during the whole duration of the project, since fully understanding the whole nature of the organization in an initial phase would be practically impossible, due to emerging concepts, change in the requirements for the meta-model and other events or new ideas that would require a reshape of previous fundamentals or concepts.

To guarantee that the metamodel defined would be flexible (so that it could allow change during the project) and accurately represent the reality of SPMS (mapping meta-model concepts to real world concepts had to be an effortless task), major decisions where taken.

First of all, we would not strictly follow previously existing meta-models, such as the one defined by Archimate. This decision was based on the lack of flexibility presented by these, since most of them

were ”written in stone”, meaning, their definition was absolute and required the following of a set of rules that might limit our capacity to accurately represent this specific context. However, due to the fact that some of the concepts are not entirely different, the Archimate notation was partly used in the modelling of the architecture. Valuable features like the use of colours to divide concepts belonging to a certain layer were useful in our representation, and as such, used. Apart from the representation part and some logical concepts, our metamodel would be defined by the ground-up to be fully committed to the accurate representation of SPMS structure. The fact that this metamodel would only be used within a ”closed” information ecosystem (SPMS and the eSIS entities), meaning the need to respect external models and/or regulations, we felt their was enough liberty to follow our own path in this issue.

Secondly, if this metamodel was meant to represent the reality of SPMS, it had to be as accurate as possible with the day-to-day concepts that existed within this organization, and above all, it had to be understood by the people that worked in it and would, in future, use it. The reason why the eSIS entities' members were left out (apart from a few select members of each entity, mostly IT administrators, that had access to official documentation) was due to the fact that these were only the end users of the systems and information produced, meaning that an in-depth understanding of these concepts was considered not necessary, since there was no practical need, for example, for a nurse of doctor to understand the difference between a solution and an application.

In order to acquire this insight, meetings and interviews with different members of the organization took place, with the author and the architectural team present, where we proceeded to try to understand what concepts the members found important to represent in the metamodel. Input from the architectural team also existed in order to study these proposals and make sure the final ”product” made stayed coherent. Like stated previously, these meetings only served to create an initial ”draft” of the metamodel, so that architectural views and information could be created in oder to support the project. New concepts have been added throughout the project that resulted from the need to represent new concepts. Restructuring of the metamodel also happened, when initially defined classes or relation no longer made sense or new one where needed.

Given this procedure, we will now describe the metamodel as of September of 2018, that resulted from this initial effort and continuous feedback.

3.1.1 Actor

Figure 3.1: Symbol used for the Actor class.

The ”Actor” class represents various concepts relating to the human part of the organization. The current version oft his class is able to represent:

- Individual human actors, like a person (Ex: Actor ”João Mendes”), although this use for the class is very limited, and used only in very specific cases that will be explored further, such as the need to properly identify the head of a department, project, direction, coordination area etc.

- The DSI (Direcção de Sistemas de Informação) as a whole, meaning an instance of the class exists that aggregates all functional departments within. This instance is unique.

- Coordination Areas (designated as Sections of the DSI), which represent the one level lower hierarchical level after SPMS. These coordination areas are responsible for the management of the aggregated departments, and have an associated ”Coordinator”, a specific individual considered the responsible for the area.

- Departments. These, like previously mentioned, are aggregated by the coordination areas group of people directly responsable for a set of projects and solutions.

- Solution profiles that are attributed to a set of corresponding actors. This type of instance will be further explored in Section 3.6

This class' instances, like those of many other classes, can have different meanings, depending on the hierarchical relationships it has with different elements and through which proprieties the ”connect” with other elements. For example, if Actor A has a connection to Actor B trough the propriety ”Director”, we know that Actor B is the director of what we assume to be a coordination area, actor A. But if instead we have that Actor A aggregates Actor B, we know that A represents something greater than Actor B, like a department or coordination area, and we can ”calculate” what it effectively based on the rest of the relations it establishes with other elements. If Actor X aggregates Actor Y, which in turn aggregates Actor Z, we can infer that Actor X is the DSI, Y is a coordination area and Z is a department. This of course requires some special attention when filling information and creating instances, since we must guarantee that these hierarchical rules are maintained.

The properties of this class are:

- Name: Unique object identifier.

- Aggregates: Actors that are aggregated by the object. For example, Coordination areas should aggregate all they departments they coordinate.

- Business Functions: In the case of the object being a ”Profile”, this property should contain all the Business Functions this Actor can perform.

- Coordinator: If the object represents a Coordination area, the property should contain the Actor responsible for it.

- Decommissioned Date: Date at which the actor becomes Decommissioned.

- Description: Brief textual description of the object.

- Director: If the object represents a Department, this property should contain the Actor responsible for it.
- Email: electronic contact information for both groups or individuals.
- Location: Geographical location of the object.
- Managed By: Actor that is responsible for the management of a certain individual.
- Phone Number: Telephonic contact information.
- Type: Textual detailed information of the role (Ex: Maintenance Provider).

3.1.2 Business Process

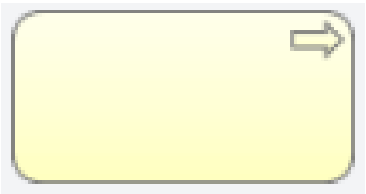

Figure 3.2: Symbol used for the Business Process class.

This class represents how the Organization realizes their Products and Services. It can be:

1. A Human Business Processes performed by Actors
2. A Business representation of an Automated Work-flow performed by an Application Service

Business Processes are usually associated with the Application services they use of the Solution that sustains them, i.e, Solutions used to perform them.

Since Business Process can have various steps and involved Actors, this class a three level hierarchy, meaning that a Business Process can aggregate other Processes, and so on, as long as this chain of aggregation does not surpass three levels. This limit was imposed so that every member of the organization was capable of synthesizing the complexity of the Business Processes, without falling into unnecessary complexity (Ex: Some members initially had five or six levels of aggregation). This allows information regarding this concept organized and within the scope of the project, since too much detail would only complicate the process of creating view and maintaining dependencies.

The properties of this class are:

- Name: Unique Object identifier
- Subprocesses: If the Business Process is divided into sub-processes, these should be aggregated by the object.
- Application: Solution or Application used to perform the Business Process.
- Application Services: Application Service used to perform the Business Process.

- Category: Textual description of the category to which the Business Process belongs (only used in specific cases).
- Decommissioned Date: Date at which the Business Process became decommissioned.
- Description: Brief textual description of the Business Process.
- IsRoot?: Boolean value that if true, means that this Business Process is one of the main ones in the organization, i.e, most relevant to business.
- Managed By: Actor responsible for the management of the Business Process.
- Productive Date: Date at which the Business Process became active.
- Data Objects: Information used and/or produced by the Business Process.

3.1.3 Business Function

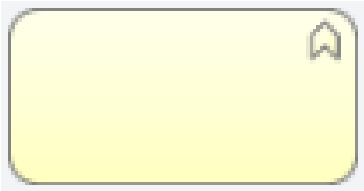

Figure 3.3: Symbol used for the Business Function class.

A Business Function in the SPMS context represents the actions that an Actor class instance (one that represents a Profile) can have over a set of information (Data Objects) in the context of a Solution. This means that we can associate and action to a profile such as a Doctor, that is also associated to a Solution, creating architectural statements such as ”within the Solution A, a Doctor can create a Clinical Report”. We establish a CRUD approach to these functions, as seen in Fig 3.4.

The sole purpose of this class is to allow the mapping of this specific information, since this was requested by the organization in order to better understand the internal permissions of access to information. This class is further explored in subsection 3.6.

The property of this class are:

- Name: Unique object identifier.
- Description: Brief Textual description of the Business Function.
- Creates: All the data information (represented by Data Object) that the Business Function creates.
- Deletes: All the data information (represented by Data Object) that the Business Function deletes.
- Reads: All the data information (represented by Data Object) that the Business Function reads (retrieves without alterations).
- Updates: All the data information (represented by Data Object) that the Business Function updates.

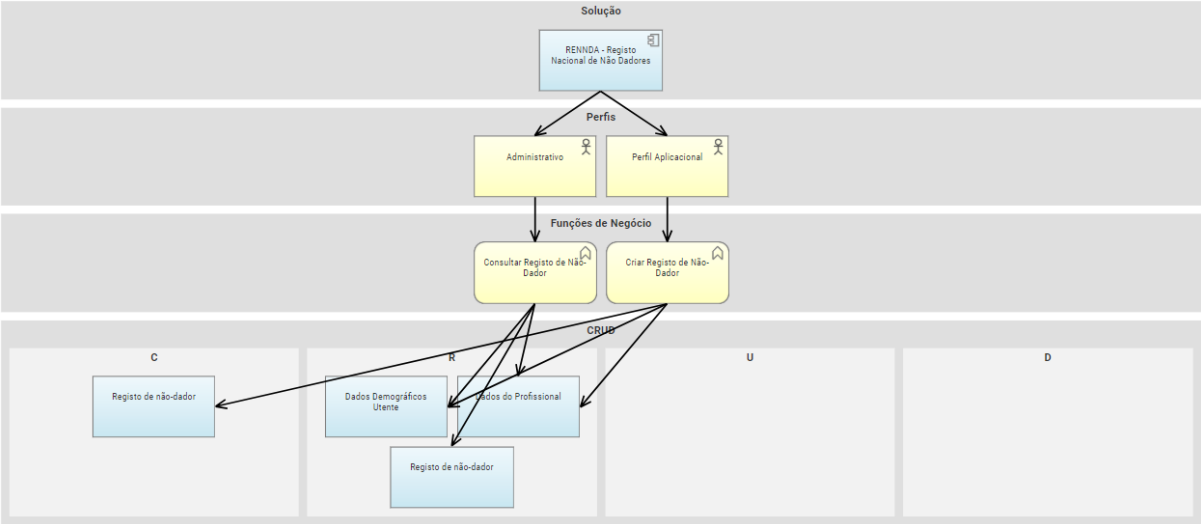

Figure 3.4: View used to visualize CRUD permissions of Profiles within a Solution. In this example, we can see that the Solution REENDA allows the existence of two user profiles (”Administrativo” and ”Perfil Aplicacional”), each with their own allowed actions (Business Functions ”Consultrar Registo de Não-Dador” and ”Criar Registo de Não-Dador”), and these allow the creation and reading of certain data, respectively.

3.1.4 Application

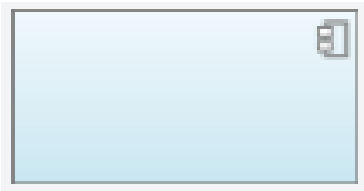

Figure 3.5: Symbol used for the Application class.

This class represents the configuration of some technology to provide Services and Functions to other Applications or to the Business. Applications expose behaviour through Application Services and use behaviour exposed by other applications. Applications may have Components that may be structured in layers or functions. Applications may have Releases that correspond to deployable instances. Only Application Releases are deployable onto a node. An Application Release may realize an application and may not realize individual Application Components. Application can be provided internally or externally to the Organization.

The Application class also follows a set of hierarchical rules built in order to better organize the concepts it is supposed to map. This division is also a consequence of the fact that technology provides the more varied set of concepts, being important to capture every single element of those.

Given this, the Application class is first separated into two classifications, the logical and physical

The logical part follows a three-level hierarchy, and corresponds to how people perceive the system, which we call Solution in this model. We do not regard the specific software implementations or needs, but instead focus on a purely functional point of view (how it serves the business).

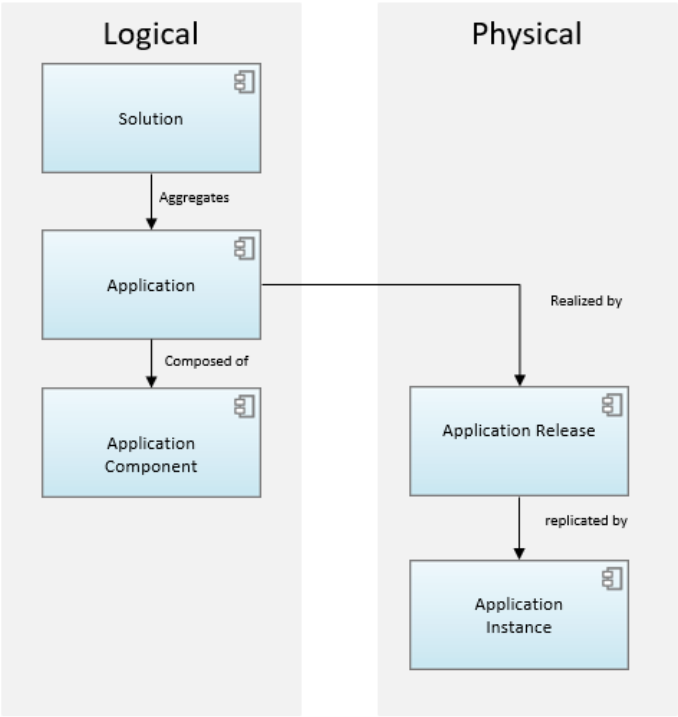

Figure 3.6: Hierarchy within the Application class.

Solution

In this perspective, we first have the concept that represents the whole. In a practical example, we have the Registo Nacional de Utentes (RNU). This is the system responsible for the distribution and registration of all the information regarding the healthcare users in Portugal. With the presented paradigm, we classify RNU as the whole. This means RNU doesn't actually exist as an entity, but rather as a logical grouping of a set of applications that work with each other for a common objective. We then go a layer a layer below, where go into further detail to describe the concepts.

Application

This second layer we put the designated "Applications". We divide these according to their functional purpose, and then classify then according to four different categories:

1. Presentation: All applications whose purpose is to provide visual presentation of the information. These can be websites that display information, or internal dashboards used by the development teams.

2. Processing: Application that function solely to treat information according to their objective and existence purpose. These can be, for example, back-office application responsible for calculations or validation of certain information.

3. Persistence/Data: Applications that are related to the storing and management of data. Usually they represent schemas or database management systems.

4. Integration: Application whose purpose is to connect/communicate/integrate with other applications through Application Services. More often than not, a lot of processing application also connect with other applications, and some solutions do not have a designated integration application. In order to simplify this issue, when representing the applications of a Solution, we group all communication with other (external) applications in an application that ”provides” or ”consumes” all the services of that solution. However, this is only done in cases where this way does not compromise (to a medium to high degree) the accuracy of the information.

With these rules, we can then group what applications make up the RNU solution: We have a visual dashboard used by the development teams to visualize the information within the BD (Presentation), we have a DBMS responsible for the management of all the information that exists within the BD (Persistence/Data), we also have a module responsible for the creation of profiles and entries using the citizen card (CC) identification (Processing). Finally we have a set of services that the solution provides to other applications to retrieve and receive information, which we group as being provided by a (non-existent) services-only application (Integration). All these modules/applications work isolated from on another, meaning that if we took it out of this context, they would still perform their tasks (given we maintain the format of the information it receives and exports). However, grouped together, they work as a whole that serves a purpose: registry of patient data from the whole country, meaning they work as a Solution. Applications can exist without the Solutions, but the reverse is not true.

Application Components

The third and final level of this hierarchical represents an analogue concept to that which is established between Solution and Applications, but with a increased level of detailed. The same way a Solution cannot exist without the Applications it aggregates, an Application cannot exist without the Application Components that it is composed of. These components represent functional modules of the applications, with a more microscopic approach (comparatively to Solutions) to detail. For example, RNU's dashboard application can be divided into different components, based on the features it provides, such as the graphs it generates on information. That feature could be mapped in a Application Component, for example, ”Map Generation Module”. Following this logic, we can decompose each application to better understand what makes it up.

Even though this decomposition into smaller components may potentially offer a more detailed vision, throughout the development of the project, most teams and departments when describing their Solutions and Applications, they ignored this last level, claiming that this amount of detail was not necessary or that useful (given the effort to map) for the visualization of the Solution's structure, opting for a more macroscopic perspective on things. This approach was understandable due to the fact that the main objective of the teams seemed to have a idea of the structure and how the solutions communicated with other in a more macroscopic manner.

Application Releases & Instances

As previously mentioned, Applications may have Releases that correspond to deployable instances and only these can be deployed onto a Node. Releases represent a version of the application, meaning it is constant in its existence.

If changes are made to the application and one wishes to deploy it onto a node, a ”snapshot” of the application is taken and a new release is created. When this version is deployed into a node, it becomes an instance of the release, meaning it exists (through the form of software) within a physical or non-physical machine.

The properties of this class are:

- Name: Unique object identifier.
- Aggregates: If the object represents a Solution, this property should contain all the applications that constitute it.
- Application Manager: Actor responsible for the management of the Solution.
- Application Type: Describes the category in which the Application fits.
- Business Functions: In the object represents a Solution, this property contains all the Business Functions that the Solution allows users to perform.
- Components: If the object represents an Application, this property should contain all the components that compose it.
- Consumes Service: In the case of Applications, this property contains all the Application Services in which the Application plays the consumer role (calls the service in the information role).
- Deployed At: For Releases and Instances, this property describes the Node where it is deployed at.
- Decommissioned Date: Date at which the Application became decommissioned (no longer part of the structure).
- Functions: Applications Functions that the Solution can perform.
- Managed By: Property that describes the Department (Actor) responsible for it’s management.
- Planned Decommissioned Date: Planned date at which the Application becomes decommissioned.
- Planned Deprecated Date: Planned date at which the Application becomes Deprecated.
- Planned Productive Date: Planned date at which the Application becomes Productive.
- Planned Under Implementation Date: Planned date at which the Application becomes under implementation.

- Under Implementation Date: Date at which the Application becomes under implementation.
- Productive Date: Date at which the Application becomes productive.
- Provider: Actor responsible for providing/selling the Solution.
- Releases: Property that lists all the releases that until a certain point in time exist and are deployed.
- Solution Typology: Property that points to the object that represents the corresponding classification category.
- System Software: Software used to support and/or create the Application.
- Used by Actors: Actors that directly interact with the Solution (use it in their daily tasks).

3.1.5 Application Service

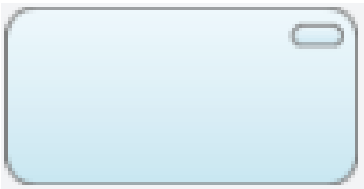

Figure 3.7: Symbol used for the Application Service class.

This class represents the behaviour performed/automated within an application and exposed to the ”outside” of it. An Application Service can only be realized by an Application (and in extremely rare cases, a system software) and be consumed by Applications, other Application Services and used by business processes.

In this context, the main goal of this class is to model generic dependencies between applications. For example, if we want to model a file transfer from Application A to Application B, this flow of information is modelled through an Application service, with one providing (realizing) the Service, while other consumes it.

This class follows the same structure as the previously mentioned Application class, meaning services are divided into Logical and Physical, with the logical being how the service is perceived in the overall structure of a Solution, and the physical represents a constant version of it (version), along with a Application Service Instance, that represents the Service deployed in a Node in the form of software. Another rule this class follows is that only applications belonging to the second-level of the hierarchy presented in Fig 3.3 (Application) can realize of consume services, never Solutions or Application Components. This was done since Solution are too generic to provide specific behaviour, and mapping service to this level would ignore too much detail. As for Application Components, these are too microscopic to provide external behaviour, since the behaviour exposed requires more complexity behind it (which translates to more components).

In order to make the maps and models more understandable, and to use the recommendations of the Archimate 3.0 metamodel, when naming a Application Service, we give a name that accurately

represents it's purpose, and not the technical name. Using the previous example, instead of calling the Application Service "DBLink123", we instead name it "File Transfer Request". This is done since we are in a level of abstraction that allows us to disregard technical details a focus on the purpose of the flow presented.

This class properties are:

- Name: Unique object identifier.
- Composed Services: Services that compose the main one, in case the service has sub-services.
- Data Objects: Information represented by the Data Object class that is transmitted and/or used by the Service.
- Deployed At: Property points to the Node at which the Application Service is deployed at.
- Description: Textual description of the Service.
- Realizing Applications: Applications that realize the services, i.e, those that provide the Service for other to use.
- Version: Lists all versions of the Service.
- Uses Software: Software used to create/support the Service.
- Planned Decommissioned Date: Planned date at which the Application Service becomes decommissioned.
- Planned Deprecated Date: Planned date at which the Application Service becomes Deprecated.
- Planned Productive Date: Planned date at which the Application Service becomes Productive.
- Planned Under Implementation Date: Planned date at which the Application Service becomes under implementation.
- Under Implementation Date: Date at which the Application Service becomes under implementation.
- Productive Date: Date at which the Application Service becomes productive.
- Decommissioned Date: Date at which the Application Service became decommissioned (no longer part of the structure).

3.1.6 Application Function

For the SPMS project, the Application Function class is used in a different manner than that traditionally seen in the Archimate 3.0 metamodel. The class still represents behaviour of an Application, but only at a Solution level. This decision was made since the objective, in the greater picture, was to understand

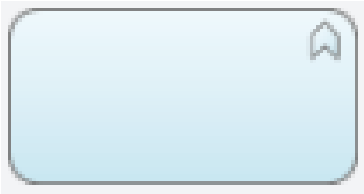

Figure 3.8: Symbol used for the Application Function class.

what each Solution could do, and the detail of what each application that a Solution aggregates was outside the scope of the modelling objectives. Also, instead of representing external behaviour that one Solution could provide to other, it also represents the level of influence or permissions a certain Solution has over information, represented in the form of Data Objects (see Section 3.1.6).

This way to map influence over information was done as the teams asked the author to find a way to represent ”what solutions use what information”. In order to fulfill this requirement, we opted to represent this structuring resorting to a CRUD Matrix(REF), which lead to the creation of the view presented in Fig 3.6.

We also choose to divide the Application Functions into two distinct hierarchical levels, the first being Macro-processes and the scond Functions. Macro-processes aggregate Functions. This was done since it was possible to group functions by a common theme or type of activity, so the creation of an aggregator made logical sense and allowed. Despite the view presenting the Macro-processes first, the Solution only directly ”connects” to functions, since directly connecting it to the Macro-processes would imply that a Solution could perform all the function aggregated, which in some cases is not true.

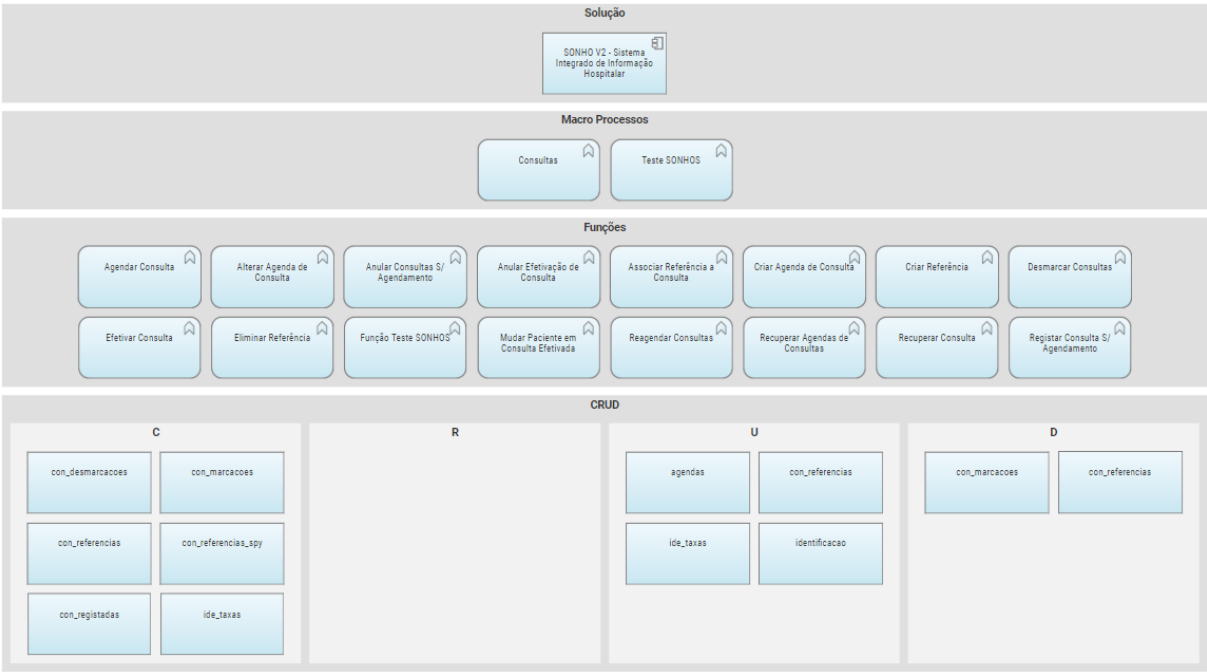

Figure 3.9: View used to expose the influence one Solution as over a set of information.

Therefore, Application Functions in this context follow a different set of rules than the ”traditional” one. This new approach allowed us to better cater to the needs of the teams, while at the same time

providing information on how Solution interact with the information they use.

This class' properties are the following:

- Name: Unique object identifier.
- Description: Textual description of the Function
- Functions: Application Functions that are aggregated by the object, in case it represents a Macro-Process.
- Creates: Data Objects that the Function creates.
- Reads: Data Objects that the Function reads.
- Updates: Data Objects that are updated by the Function.
- Deletes: Data Objects deleted by the Function.
- Supports: Property that points to the Business Function which the function supports the execution (can be multiple).

3.1.7 Solution Typology

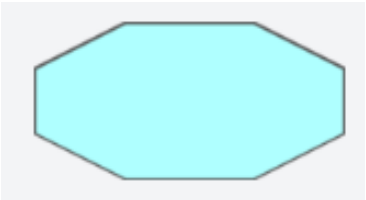

Figure 3.10: Symbol used for the Solution Typology class.

This class represents the classification that is attributed to a Solution according to the standards defined by the SPMS. At the time of writing, no final decision on the categories that would exist had happened. However, this class exists and is ready to be represented and used for the classification of Solutions.

This class originated from the need to represent some concepts related to the General Data Protection Regulation (GDPR) imposed by the European Union. The factors behind this classification range from a variety of themes, such as access rights, security, information used, Node where it exists, sustainability and business purpose.

The properties of this class are the following:

- Name: The object's unique identifier.
- Description: Textual description of the object

- Category: Textual property that may or may not be used to further discriminate the instances of this class.
- Sub-typologies: In case some typologies represent an aggregator of smaller typologies, this field should list all the Solution Typology aggregated by the object.

3.1.8 Data Object

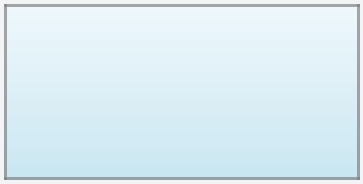

Figure 3.11: Symbol used for the Data Object class.

The purpose of this class is to represent information. More specifically, this class represents the information that goes through, is used by, created, updated and deleted by the concepts represented by the class Application. Data Objects can be associated to Business Functions and Application Functions, using the already explored CRUD paradigm, and can also be associated to Business and Application Functions, using the CRUD paradigm, Actors that have a very important relation with them (this case is of very rare use, but for example, we associate the Data Object ”Clinical Report” to the Profile Doctor, since only a Doctor can create this object, through a system), as well as Application Service, describing the information that goes through that Service.

This metamodel approaches this class in a very generic way, meaning that a Data Object can be used to map a plethora of elements, that range from the age of a patient to a full clinical document. In order to overcome redundancy or lack of detail, we name data objects in a way that literally represents what they are supposed to mean. For example, if Application A sends a clinical report to Application B, we name that Data Object ”Clinical Report”. On top of this, context becomes a very important part of the use of this class. One of the reasons being this generic approach is that teams only wanted to know, in the context of communication between their Solutions with others, the information that went from one place to another, with disregard for the technical details behind that information. It was assumed (with the approval of the members) that when a team wanted to analyze what goes from one place from another, they only needed to know ”what”, without information of the specific data format or how this data was treated, as those details were out of the scope.

This class’ properties are:

- Name: Unique Object identifier.
- Data Objects: Lists the Data Objects that compose the Object, if it contains
- Description: Textual description of the information contained within the Data Object.

- Data Typology: Property that points to the object that represents the corresponding data classification.
- Planned Decommissioned Date: Planned date at which the Data Object becomes decommissioned.
- Planned Deprecated Date: Planned date at which the Data Object becomes Deprecated.
- Planned Productive Date: Planned date at which the Data Object becomes Productive.
- Planned Under Implementation Date: Planned date at which the Data Object becomes under implementation.
- Under Implementation Date: Date at which the Data Object becomes under implementation.
- Productive Date: Date at which the Data Object becomes productive.
- Decommissioned Date: Date at which the Data Object became decommissioned (no longer part of the structure).

3.1.9 Data Typology

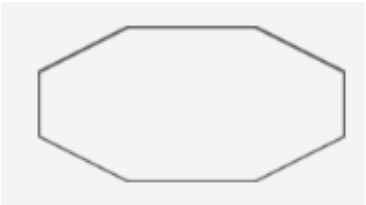

Figure 3.12: Symbol used for the Data Typology class.

This class serves the exact same purpose as the Solution Typology, except it is applied to Data Objects. It represents the classification that is attributed to a Data Object according to the standards defined by SPMS, according to the need to map GDPR concepts. The factors behind this classification range from a variety of themes, such as access rights, security, information used, Node where it exists, sustainability and business purpose.

This class' properties are:

- Name: Unique object identifier.
- Category: This field exists in case the classification becomes more complex and additional subclassifications need to be created.
- Description: Description of the instance of the class.

Figure 3.13: Symbol used for the Environment class.

3.1.10 Environment

The Environment class serves as logical grouping class that aggregates a set of Nodes that share a information or purpose with a very high level of dependency. For example, to map the SPMS testing Environment, an instance of this class is created, and we associate all the logical and physical Node class instances used to support the testing of software or technology related efforts.

This class serves a more organizational role, as it's purpose is to organize Nodes according to their purpose within the technological layer for the organization, and facilitate the lives of the development teams.

This class' properties are the following:

- Name: Unique object identifier.
- Environments: In case an Environment is composed by several other, this property should list the sub-environments of the object.
- Description: Textual description of the Environment.
- Managed By: Actor responsible for the management of the Environment.
- Nodes: Points to all the Nodes that belong to the Environment.

3.1.11 Node

Figure 3.14: Symbol used for the Node class.

The Node class serves the exact same purpose as the one defined in the Archimate 3.0 metamodel, representing a computational resource upon which Applications Releases and System Software are deployed. Nodes also have a given computing capacity (MIPS, RAM, DISK, Network, Availability, Scalability). Nodes can also exist as physical identities (Physical Server), Virtual Machines or Cloud Servers. This differentiation is done through properties of the class.

Nodes can aggregate others Node, and so on. Unlike other classes that can aggregate themselves, we did not establish a ”limit rule” of how many layers of aggregation can exist. This is due the fact that in this specific case, the layers of aggregation vary a lot, and imposing a limited, even for the sake of simplicity, would result in inaccurate representation of how things existed, so along with the teams, we opted to give a higher level of liberty in this aspect.

This class' properties are:

- Name: Unique object identifier.
- Capacity: Storage capacity of the Node (in GB).
- Depends On: Nodes with whom the object as a high dependency degree in terms of proper functioning.
- Description: Textual description of the Node.
- Environment: Property that points to the Environment to which the Node belongs to.
- IsVirtual?: Property that describes the Node in terms of physicality.
- Location: geographic location in which the Node exists.
- Maintenance Provider: Actor responsible for the Maintenance of the Node.
- Managed By: Actor responsible for the management of the Node.
- Manufacturer: Actor responsible for the manufacturing of the Node.
- Nodes: Lists all Nodes that compose the object.
- System Software: Software directly installed in the Node that supports it's functioning and purpose.
- Node Typology: Property that points to the respective classification attributed to the Node.
- Planned Decommissioned Date: Planned date at which the Node becomes decommissioned.
- Planned Deprecated Date: Planned date at which the Node becomes Deprecated.
- Planned Productive Date: Planned date at which the Node becomes Productive.
- Planned Under Implementation Date: Planned date at which the Node becomes under implementation.
- Under Implementation Date: Date at which the Node becomes under implementation.
- Productive Date: Date at which the Node becomes productive.
- Decommissioned Date: Date at which the Node became decommissioned (no longer part of the structure).

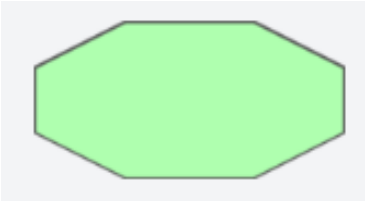

Figure 3.15: Symbol used for the Node Typology class.

3.1.12 Node Typology

This class serves the exact same purpose as the Solution Typology, except it is applied to Nodes. It represents the classification that is attributed to a Node according to the standards defined by SPMS, according to the need to map GDPR concepts. The factors behind this classification range from a variety of themes, such as access rights, security, information stored, sustainability and business purpose.

This class' properties are:

- Name: Unique object identifier
- Category: Field created to support extension of the class, may the need arise.
- Description: Description of the instance.
- Sub-typologies: Following SPMS classification needs, there exist two types of classifications for Nodes, the macro and the sub. Macro classifications aggregate sub-classification, which in turn are directly associated to nodes

3.1.13 Location

Figure 3.16: Symbol used for the Location class.

This class represents a physical or non-physical location where Nodes can exist. In the case of a physical location (and thus a physical Node), we associate the Node to the instance of the class Location that accurately represents reality. In the case of a non-physical Node (ex: Cloud Server), a instance of a location is also created, but this time with the name of the cloud service where the cloud server exists.

This class' properties are the following:

- Name: Unique object identifier.
- Address: Physical address of the location.

- Description: Textual description of the Location.
- Aggregates: If the Location has several sub-locations where Nodes are deployed, this property points to those same sub-locations.
- Geo: Geographical Coordinates of the location.
- IsVirtual?: If the location is non-physical (ex: cloud) or physically exists.
- Managed By: Actor responsible for the Management of the location.

3.1.14 System Software

This class represents all software that is acquired from suppliers and whose management, roadmap and availability is ultimately out of control of the organization. As Applications, System Software have a deployable instance (releases) and a non-deployable instance (logical).

Figure 3.17: Symbol used for the System Software class.

System Software is usually associated to the application it supports and/or helps create, meaning that if for example, an Application in the Presentation category uses a specific technology, such as Angular, we create an instance that represents that software and associate it to the respective Application instance. Software can also be associated to Application Services, in case that specific service uses a certain technology, and also to Node. In the case of Nodes, we do not map directly to the Node the technology used by the Applications, instead Application Instances that are deployed into a Node have the technology they use directly associated to them, and by logical conclusion, we conclude that said technology also exists within the Node. If we are describing technology that is only used by the Node, such as Database Management Systems, then we directly connect that technology to the Node.

This class' properties are the following:

- Name: Unique object identifier.
- Deployed At: Node(s) in which the Software is deployed at.
- Description: Textual description of the Software.
- Development Manager: If the the software is first party, the Actor responsible for the management of the development should be in this property.

- Managed By: Actor responsible for the management of the Software. This differs from the previous property since the software can be purchased (third-party), and someone at the Organization becomes the responsible for it.
- Provider: External provider of the Software.
- System Software Releases: Existing releases of the software.
- System Software Code: If necessary, teams can use this property to write purposeful information regarding the code of the software.
- Availability: Provides the expected availability of the code.
- Planned Decommissioned Date: Planned date at which the System Software becomes decommissioned.
- Planned Deprecated Date: Planned date at which the System Software becomes Deprecated.
- Planned Productive Date: Planned date at which the System Software becomes Productive.
- Planned Under Implementation Date: Planned date at which the System Software becomes under implementation.
- Under Implementation Date: Date at which the System Software becomes under implementation.
- Productive Date: Date at which the Node becomes productive.
- Decommissioned Date: Date at which the System Software became decommissioned (no longer part of the structure).

3.1.15 Project

Figure 3.18: Symbol used for the Project class.

As referenced in [37], Projects represent a transformation initiative with a clearly defined beginning and termination dates, and a well-defined set of goals or results. Every purposeful change in the organization is associated to a project, such as for introduce a new productive EA artifact in the Architecture or the decommission of EA artifacts. An example of a non-purposeful change is technology obsolescence.

Projects can be associated to a set of objects, according to the influence the project has over them. This class has the following properties:

- Name: The unique object identifier.

- Change: When the organization just wishes to know what is involved in this project, without the detail of how it is changed, this property allows the connection between the project class and all other existing classes.
- Coordination Area: The Actor Class instance that represents the coordination area responsible for the project.
- Decommission: Lists all objects from whatever class that are decommissioned at the end of the Project.
- Productive Date: Date at which the Project is planned to begin.
- Decommissioned Date: Date at which the Project is decommissioned (ends).
- Description: Brief textual description of the Project.
- Managed By: Actor responsible for the management of the Project (Project Manager).
- Productive: Lists all the objects that become productive upon the realization of the Project.
- Projects: In case of a large scale project, sub-projects may be created, and should be associated to the main one.
- Read: Lists all objects from which the Project must take information from.

3.2 Utilization of Forms to Streamline Information Collecting

In sections 1.2.1 and specially 1.2.7, the themes of effort and information collection were discussed. We reached the conclusion that manually importing information to a certain repository proved to be a very cumbersome task. In the beginning of the project, the Atlas tool only supported automatic importation of information through pre-treated Excel files or XML files, and from manually inputting information through a very cumbersome process, in which the user had to manually create instances, search from them in the instance browser and manually edit each of the properties, one by one. This of course became problematic, since creating Excel and XML files for the automatic importation required some understanding of the formats used by the Atlas tool, and implied that every time some team members wished to objects, they had to fully edit a file to be imported. The manual alternative was also time consuming, had needed users to have some understanding of the properties, relations and overall metamodel being used. .

It was then necessary to find a way to make information collecting and editing a more seamless effort than ever before, since not only would it speed up the process of enriching the repositories with information, but also allowed for a more positive and less complicated experience for the users.

In order to achieve this goal, we decided to extend the tool's features and come up with a way to fill information in such way that a deep knowledge of the metamodel was not necessary, being only necessary that the users had a surface-level understanding of the metamodel. Through this abstraction, we hid technical details from the user that represented a roadblock for them to perform the simple task of filling he right properties and creating the right classes. This new feature are the Forms.

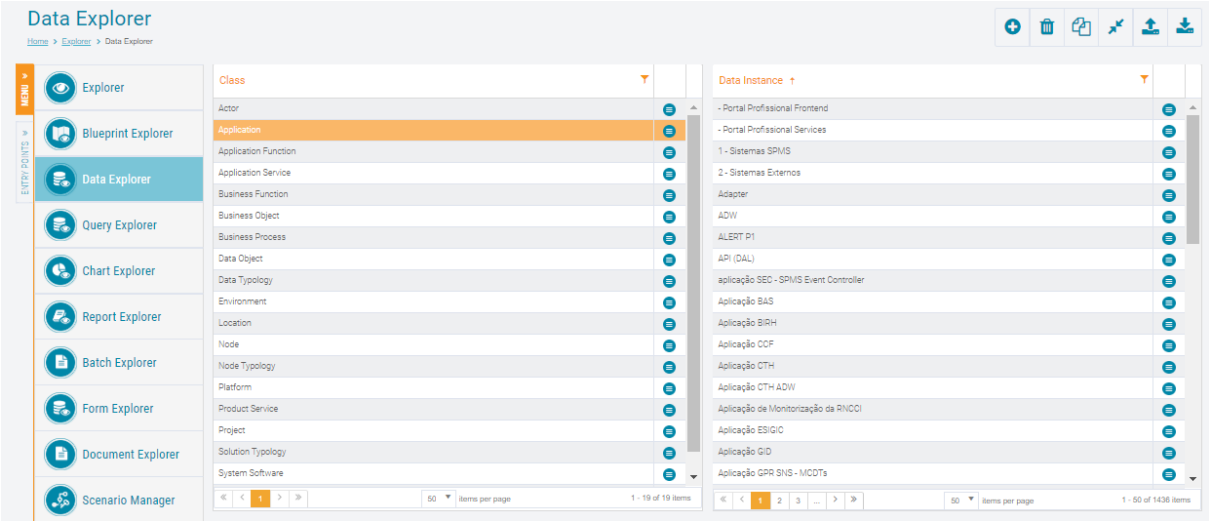

Figure 3.19: To manually input information, users had to constantly use this screen, choosing different instances and classes to make sure everything was properly related.

3.2.1 Forms

Atlas' Forms were the new feature developed to tackle the problem of cumbersome information collecting and to make the lives of the organization's members easier in that aspect. Instead of manually searching for the instances and creating them, the users only need to click of the Form corresponding to the type of object they wish to create/update.

This feature presents itself literally as a form that presents the user with a set of empty spaces that correspond to a certain property of the class being created. Some spaces are obligatory depending on the type of object being created. This abstraction helps users better fulfill their tasks. By removing the technical aspect of having to know how to access a singular instance, properly edit it's properties in the right way, Forms present an extra layer that "translates" the previous effort into a more seamless endeavour, using more "layman" terms. For example, when asked what applications are part of the Solution being created, the user does not need to know that the property "Aggregates" is the one used, and that objects belonging to that class are become connected through an "aggregation" relation.

Upon filling the form, the user must save the changes applied to the object, and Atlas will create the instance. This, in theory, presents a much faster alternative to the creation of objects, although it does still require a form to be filled for each object that needs to be created.

Users can also edit existing objects with this feature. Given the correct form is selected (ex: if one wishes to edit a Actor object, the Actor form must be selected). After accessing the form, one must simply input the name of the object it wishes to edit (a drop-down list assists the user to prevent misspelling), and edit the respective property. Not all properties follow the same update rules. For example, the description as a destructive update rule, meaning that whatever new value that is inputted when updating is now the only value that property has. The Application Components for an Application on the other hand, have an additive update, meaning that any value inputted in the update will now be part of the previous ones.

These decisions were made based on the feedback from the teams and the review of how organiza-

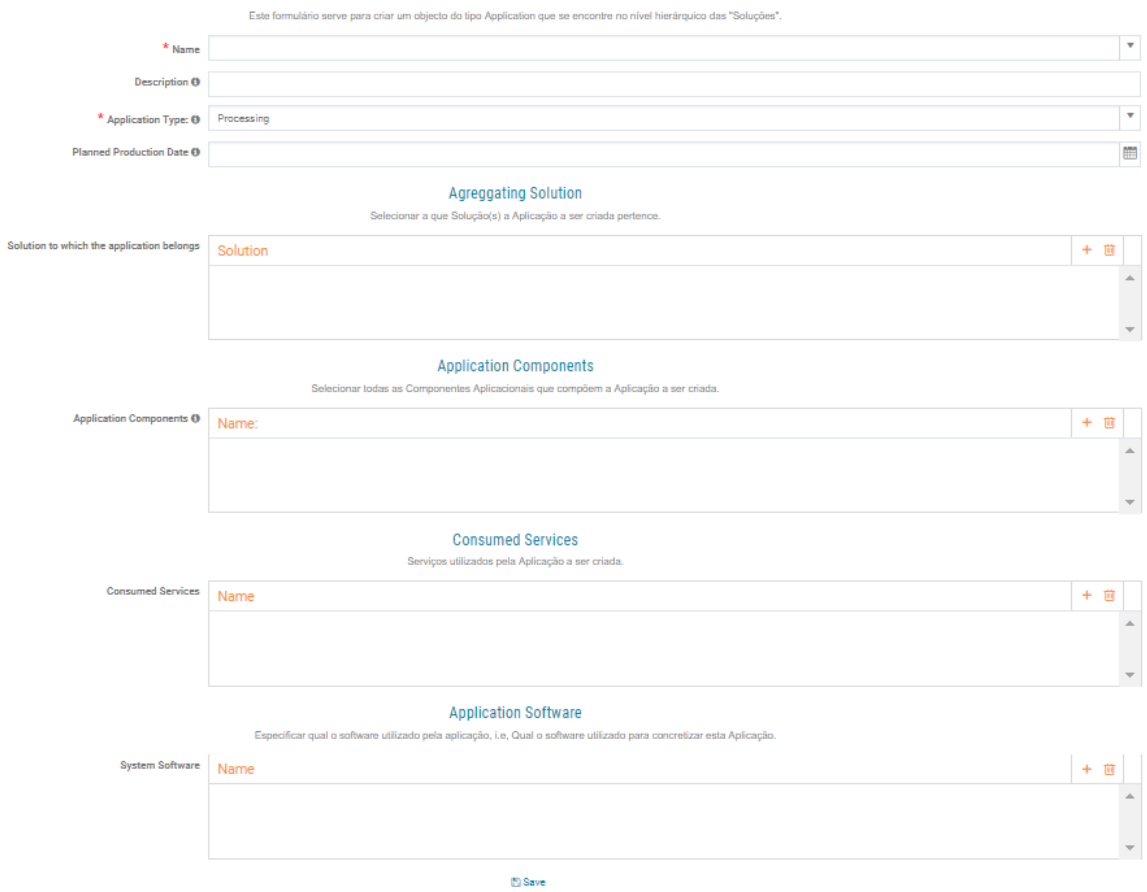

Figure 3.20: Example of a Form (Application Register). This form allows users to create an Object of the Application that represents an application in the hierarchical structure of the class.

tion members reacted to the old manual input method. Logistical reasons also played a big role, since, for example, having the users input all the Application Components every time they wished to add a single one was counter-productive.

3.2.2 Configuring Forms

In order to provide the necessary Forms according to the needs of the organization, one must first configure them. The development of the tool itself was not handled by the author, but by a group of software engineers from Link Consulting. This features allows now only Forms to exist, but also for users (with the correct permissions) to configure new forms, may the need arise.

To create the necessary forms, the author, along with the EA team of the Organization, defined the most needed forms, according to the objectives for the overall project. Since the main goal was to completely map the information systems layers, the actors that interact with it and the infrastructure, a list of forms was defined:

- Solution Register: Allows for the creation of an Application Class object that represents a Solution.
- Application Register: Allows for the creation of an Application Class object that represents an Application.

- Application Component Register: Allows for the creation of an Application Class object that represents an Application Component.
- Application Service Register: Allows for the creation of an Application Service object.
- Individual Actor Register: Allows for the creation of an Actor object that represents an individual.
- Application Function Register: Allows for the creation of an Application Function Class object.
- Data Object Register: Allows for the creation of a Data Object class object.
- Node Register: Allows for the creation of a Node Class Object.
- Business Function Register: Allows for the creation of an Business Function Class object.

The effort to configured each Form took some initial time to understand how the feature works at a technical level, and upon having that knowledge, the time spent configuring one depended on the complexity of the information it required. For example, the Form responsible for the registration of an Actor class object that represents a hierarchical Unit is relatively simple, since only the name, manager, hierarchical unit that it aggregates or is aggregated by was needed. However, for Application Services and Applications, more information was required, such as the System Software, the components that make up these etc. This translated into a more complex Form to configure, taking an average time of somewhere between 15 to 45 minutes to configure, plus testing time (about 10 minutes, depending on the correctness of the initial implementation) and along with the importation of these Forms to other eSIS repositories.

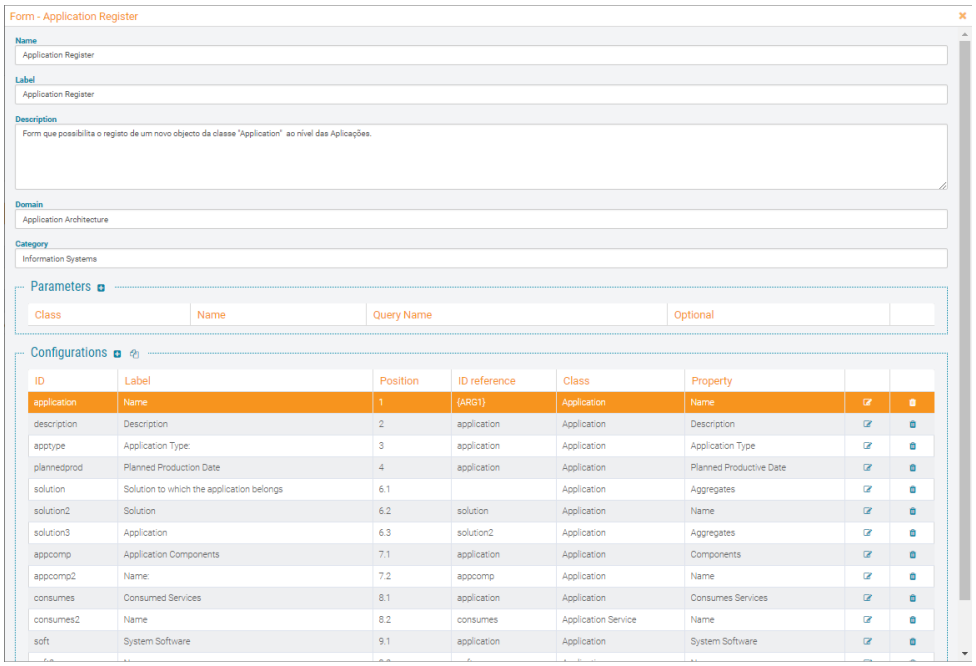

Figure 3.21: Screen for the configuration of Forms. In this example, we have the configuration for the Application Register Form.

The reason why some concepts were left out of the Forms, such as the Location or the Actors that represent hierarchical units, is that most of these did not compensate the creation of a Form. For

locations for example, they are a constant, locations will exist regardless of structural organization, and can be created manually or through the automatic importation of information to the platform, making it a one time effort. Creating a Form for this specific class will result in a Form that would only fulfill a very-short lived objective, and would end up taking space in the menu presented to the user, to never be used again or used with very low-frequency.

The technical aspects of the configuration are not the focus of this subsection, since the objective is for the reader to understand that an abstraction is presented to the users, meaning only those responsible for the creation of the forms must have a deep understanding of the meta-model, how each properties relate with each other and what buttons to press when configuring a Form, and that technical implementation is not relevant for the topic at hand.

3.3 Separation of Design and Representation

SPMS was founded in 2010, and as such, much of the planning for the architecture, systems and other concerns were already in place when the project described in this thesis began to take place. The goal of the project at hand is not to conceive the most optimal enterprise architecture or give advice in how to construct one, and was never approached as such. As previously mentioned, the sole purpose of this project is to organize existing information, hierarchies and structure in such way that it can become easily ”readable” and help the organization it it's endeavours.

Following the methods described in [14] and [37], we approach this project entirely as a Enterprise Cartography project, meaning that a set of principles and rules on how to act and solve issues heavily based on the principle of separation between Design and Representation was followed, focusing on creating a feedback loop between what is represented in the views constructed using retrieved information and the changes the analysis of these views (and other information) cause.

Firstly, we had to clearly define what the distinction between Design and Representation meant in the context of the presented project. Following the definition mentioned in [14]:

- Purposeful Design: Regards the shape of the systems, concerns changes to the existing architecture and how these should exist.
- Cartography (Representation): Purely descriptive, has no regard for the Design. Fully committed on the visual description of the existing information.

This translated into an approach where the main focus should only be the creation of views/maps that accurately represented reality, without questioning if the design. This allowed the architectural team to have a clear idea of what was intended, use this idea to better focus on the objectives.

Upon deciding the type of project, the author followed the template for EC projects exposed in section 2.4, as it presented detailed steps for every phase of the project, as well as practical examples of this type of implementation.

As such, the following steps were taken during the implementation of the project:

1. Identify Project Goals: The main goal for the EA project at SPMS is to map the existing information systems, Infrastructure and Business Processes (ordered by importance). The organization wishes to have a way that allows it to intuitively visualize the information that makes-up the whole structure within it.

2. Define the meta-model: As described in it's entirety in section 3.2, a new meta-model was built from the ground up, using a set of already described methods. This model will allow the architectural team to find a "language" to represent the information within the organization and be understood by its members.

3. Identify the best sources of information: In order to represent the reality of an enterprise, we must first retrieve the information necessary for that representation. As previously mentioned, the main sources of information used were the Excel and XML files extracted from existing documentation, as well as the organization members input directly into the architectural tool being used. For the eSIS entities, the Forms and Excel files were the main source of information.

4. Structure the Processes and Tools to Capture Information: As previously mentioned, two main sources of information were established: the Excel and XML files provided by the different entities (SPMS and eSIS) and members of the SPMS organization, and the forms members filled. A schedule for the retrieval of information was also established, with for example, in a determined week, a specific entity was asked to provide information regarding their infrastructure in a week's time. This was done continuously throughout the project's duration.

5. Define and Configure Architectural Maps: In order to fulfill the needs of the members of the organization, these were inquired about the views they wished to have of the information collected in the Atlas tool, and using the tool's capabilities, these maps were created. The following subsection will focus on the maps produced for this project.

3.3.1 Architectural Maps

The information contained within the architectural repository, in order to achieve a greater level of purpose, must be exposed in the form of views. These views were developed according to the needs of the organization. By talking to members of the organization from different hierarchical levels, we were able to understand what needed to be represented in visual form, as well as how it should be presented.

Each map has a certain level of complexity, that varies with the information needed to create it and how it is structured. The time spent on creating maps took normally from 5 to 45 minutes, depending on the aforementioned complexity.

Maps are divided into two categories: Organic and Structural. Organic maps are simpler in their construction, and follow a Root-leaf principle, meaning a set of objects (or one passed as an argument) is used as the root, and the leaves are reached through the relation they establish with their parent node. Structural view are more complex in nature, and require the use of queries to retrieve all the information that one wishes to visualize.

3.4 Use Atlas to Support Management

One of the goals of this project was to find a way for the management of SPMS and eSIS entities to use the architectural information to support their managerial tasks. Management is used as a broad term, since in the context of this project, it includes various types of management. Using the features available in the ATLAS tool, we found ways to present information to support these managerial tasks, using both views and Charts.

Management can range from the holistic perspective of a top management officer (higher hierarchy levels) within SPMS, that wish to know how many stakeholders are involved, what providers of a certain product exist etc, to infrastructure workers that want to inform themselves on the structure of the nodes, the software that is deployed in them, the actor responsible for their maintenance etc.

On top of that, the fact that these maps are generated using queries that run every time the user wishes to visualize the view, we can guarantee that maps do not become obsolete. Since queries impose a set of constraints on information so that only what is pretended in presented in the view, if new information is added to the repository, and that information respects those constraints, there is no need to manually change the view, since the query will automatically take this new information into account.

Besides the views feature, in order to fulfill the needs of the organizations to a higher degree, we also introduced the Charts feature. Charts allow the user to visualize meta-data regarding the information present in the repository presented in a statistical manner. This feature will be further explained in the following subsection.

3.4.1 ATLAS' Charts

This ATLAS feature allows the user to visualize information in the form of coloured charts, as presented in Fig 3.19. The information presented on said charts can be configured, since these work in a similar way to the views in terms of how the information is retrieved from the repository, through the use of queries.

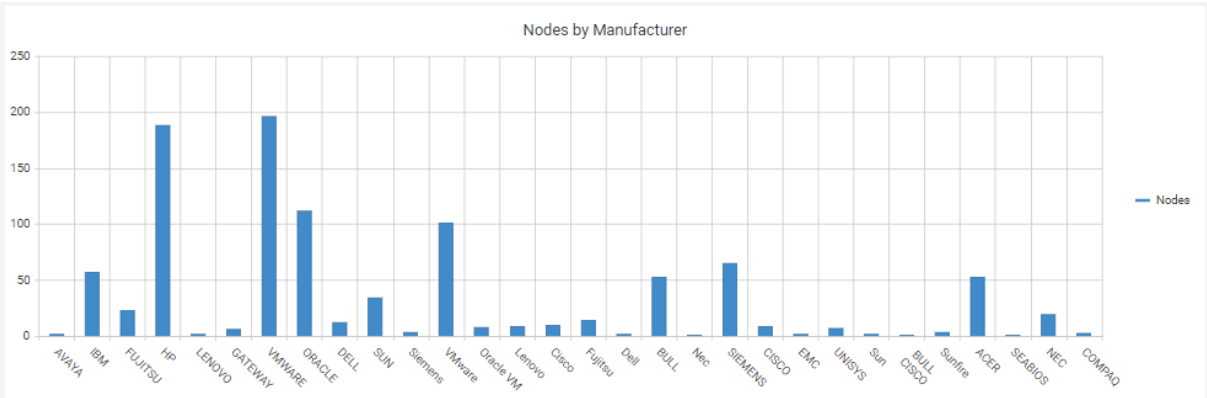

Figure 3.22: Example of a Chart. This one presents the number of Nodes by the Actors responsible for manufacturing them in the whole eSIS ecosystem.

The configuration of a Charts is a relatively easy task, however, it should only be performed by

someone with a solid technical insight of how the meta-model is structured and how the query system works, meaning a regular ATLAS user may not be able to configure a chart on his/her own.

Charts can serve as a supporting tool for the management of the real life concepts mapped within the repository, since it allows the visualization and analysis of meta-data, such as the number of providers, node maintenance providers etc, as well as the management of repositories within the tool (this topic is further explored in section 3.8).

The purpose of this feature is not to be an alternative to traditional management methods (due to it's limitations), but instead to support the managerial tasks, since it gathers information about a large quantity of data in a single chart (at the expense of some detail).

3.4.2 Practical use of Charts and Maps for Management

Even if the idea behind this feature is sound, it's success depends entirely on the information existent within the repository. In order to convince the organization of the viability of using maps and charts as a tool to support management, a pilot test was created. To prove the usefulness of this solution, the architectural team meet with the team responsible for the management of SPMS' and eSIS infrastructure to understand their needs. One of the greatest roadblocks for the team was the scattered information about nodes and their deployments. Without individually contacting each entity when needed, the team had no clue of how the infrastructure was constructed, where it was deployed, what software was deployed in each node and what and who were the node's manufactures and maintenance providers. ATLAS provided a solution to the one of the initial problems: a single platform where all this information could be stored. But this wasn't enough, since browsing this information manually would still be a time-consuming task. This leads to a very unfortunate situation, as explained by one of the members, if for example a Node has a problem, the entity calls the infrastructure team to notify the problem, and the information of the failed node has to be manually searched by the team, using loose documents and phone calls to the IT managers of the entity, to understand how the node positions itself in the hierarchy, the dependencies with other components, the actors that have responsibilities for the node etc.

In order to avoid situations like these, the map "Node Organic" was created.

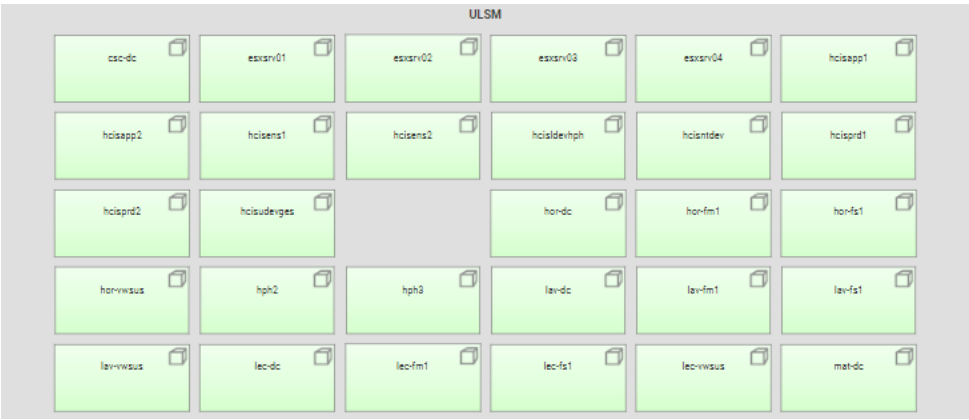

Figure 3.23: Node Organic view. Presents all the Nodes that exist within an Entity.

This view lists all the Nodes that exist within an Entity. The eSIS repository version differs in the fact that it lists all the entities before presenting the respective view in Fig 3.20.

This view represents a gateway to a more detailed map, regarding a specific Node. Upon having all the Node listed, the user looks for the Node he/she pretends to analyze. By clicking on the respective Node, the user is transported to a new view, ”Node Context”.

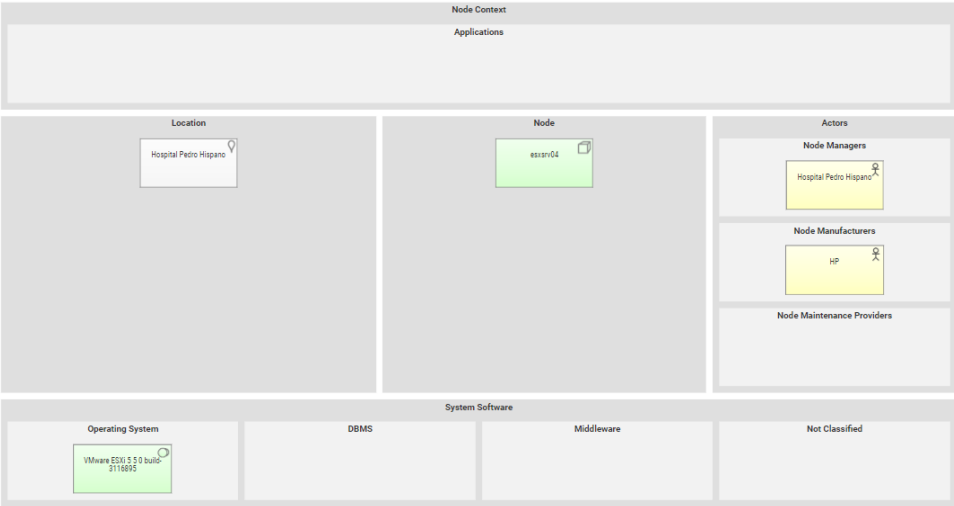

Figure 3.24: Node Context view. Presents the Node, and other objects that in some way establish a relation with it.

As the name suggests, this view presents the context in which the Node exists, where it is deployed, the actors that are responsible for it’s maintenance and manufacturing, the applications that are deployed in it and all the software it uses to function. The view presents all the basic information the user needs to know about the argument Node. If the user wishes to get further detail from the node, it can directly access the Node’s properties, and check, for example, the IP Address (its only shown to users with the correct permissions, due to security concerns), the Node aggregate it or it aggregates. Information about the other objects, such as the location can be explored in the same way (through the properties), if for example, the user needs to know the address of the deployment location or the phone number of the Node’s manufacturers.

On top of this view, charts like the one presented in Fig 3.19 allows those responsible for the management of the infrastructure to obtain the information they need to perform their tasks in a more direct and faster manner.

The same logic was applied to other contexts, such as the solution development teams. As stated before, knowing the dependencies between applications and other objects (including other applications) was one of the most important requests. A context map similar to the one presented before was created, ”Solution Context”

This view allows for the user to see what Business Processes are supported by the Solution, the Solutions from where the Solution retrieves information (provider solutions) and those to which the Solution gives information to (consumer solutions). The nodes at which the Solution’s applications are deployed are also present, as well as the software that supports the existence of the Solution’s appli-

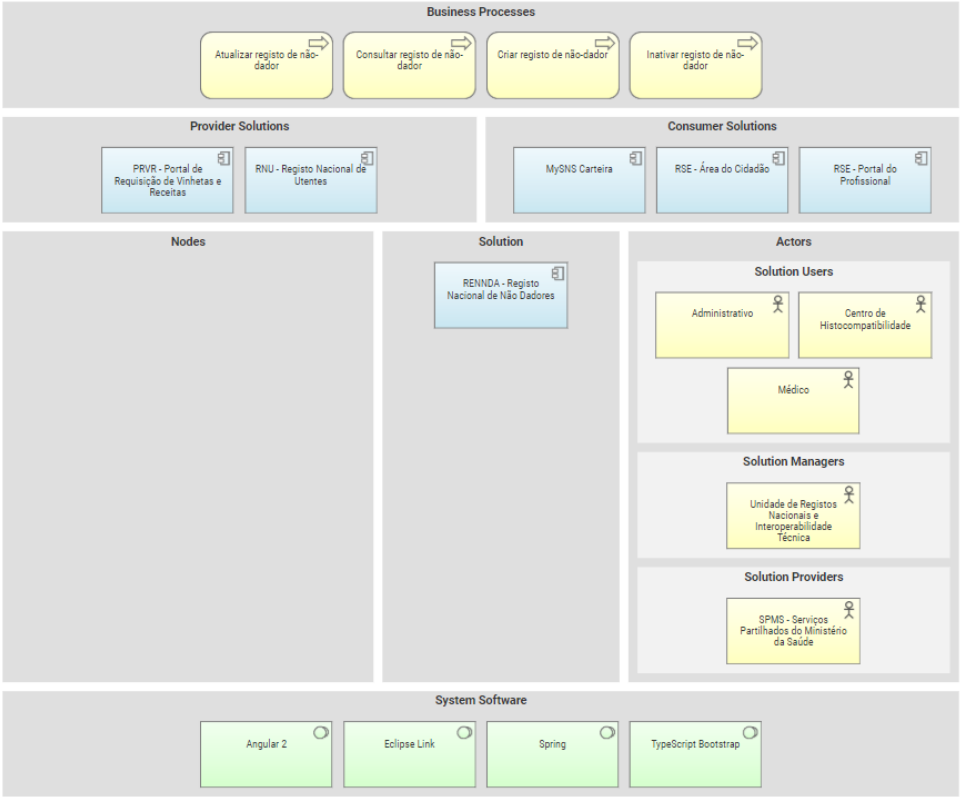

Figure 3.25: Solution Context view. Presents the Solution, and other objects that in some way establish a relation with it.

cations. If the user wishes, it can also navigate to other application class-themed views, namely the ”Solution Integration” view, one of the most important maps in terms of describing the relations between Solutions.

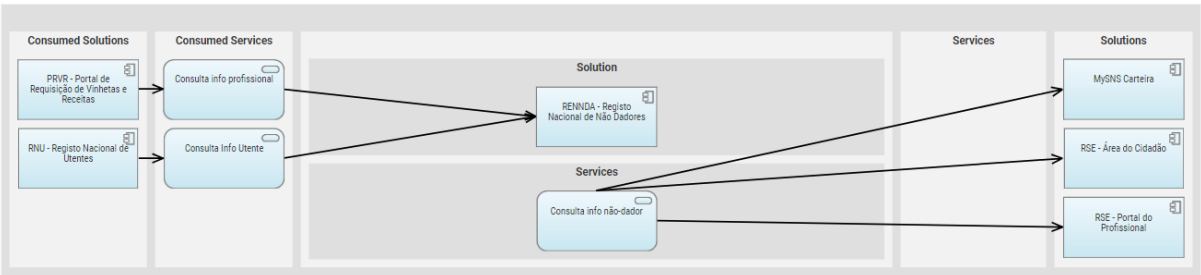

Figure 3.26: Solution Integration view. Presents the way the solution communicates and transfers information with other Solutions through application services created and used by the Solutions.

The Solution Integration view allows the user to understand, in a more general perspective, how the argument Solution communicates with other solutions, and through what services that communication happens. This map provides a useful insight for teams to trace information flows and understand how communication protocols are structured, and in case of failure of one of the components, be able to trace the problem to it’s possible origin, therefore making the task of managing communications between solutions an easier task.

Finally, teams also needed to understand the structure of a solution. While the meta-model specifics

can be used to describe these hierarchical relations, without a proper visual representation, understanding it can be difficult. To overcome this and fulfill this requirement. The view ”Solution Structure & Deploy” was created.

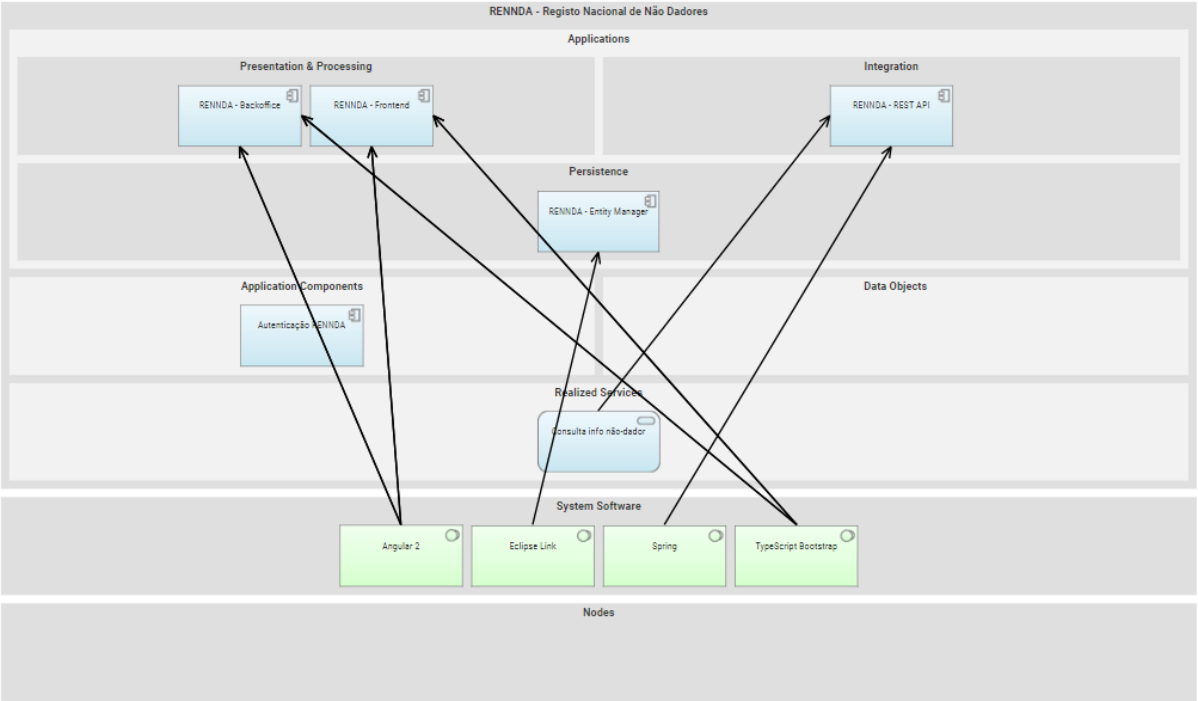

Figure 3.27: Solution Structure & Deploy view. Presents the internal structure of the argument solution.

This view allows users to visualize the structure of a Solution, according to the information that describes it present in the repository. This map includes all the applications that the Solution aggregates, categorized according to the already mentioned application classifications. On top of that, the application components of said applications are also exposed, although these are not as organized as the applications since many of the teams found that reaching that level of detail (description of the components) was not necessary for the understanding they were trying to get from the map. Nevertheless, in order to not obscure this information, it is still presented in the map, but not much attention is given to it. The services the applications provide are also listed, and by clicking the ”relationships” button, an arrow between these points the service to the respective application. Users can also analyze what software is used to support/build the respective applications, as well as the Nodes in which these are deployed at.

This view help the development teams to better manage their solution by presenting all the different components that make up their solutions, allowing a more informed management and planning.

3.5 Use Atlas to support Planning

One of the main problems with static architectural maps is their inherent ephemeral nature, not only in the context of the subject organizations of this thesis. A map built at a certain point in time uses the information that existed at that same time the map was created. The problem arises when reality changes, and so should it’s representation, but since this is one is static, a new one has to be created

or previous ones have to be updated, which becomes problematic in an organization like SPMS that changes with very high frequency (in 2018 alone, 16 new projects to create and/or update Solutions where planned and approved). Teams demanded that a solution to this problem was implemented within the ATLAS architectural tool.

Since time is how we measure change, and the organization's projects measure themselves with schedules and deadlines, we introduced the concept of time to architectural maps. In truth, time became a property of objects, in the form of dates. We may not be able to monitor in real-time how the Business Processes, Actors, Information systems or Infrastructure change, but we know that certain dates have a meaning to these, for example, we can know that solution A become productive at a certain date, and that it will be decommissioned at another, according to the schedules defined by the organization. With this information, we can then visually represent these dates and how they changed the state of the objects, as explained in [14].

Currently, as already demonstrated in the meta-model description, seven dates exist:

- Planned Decommissioned Date
- Decommissioned Date
- Planned Deprecated Date
- Deprecated Date
- Planned Under Implementation Date
- Under Implementation Date
- Planned Productive Date
- Productive Date

Planned dates refer to the dates that were originally proposed as the dates at which the objects would change states, while the other dates refer to the actual date at which those changes occurred. These planned dates were registered as they allowed users to be aware of the delays or compare dates to decide, if the need may arise, what dates could be pushed or changed. Having a central platform with all the original dates is also relevant, since users wanted a way to consult dates without having to resort to asking for official documentation individually from various sources, guaranteeing that everybody had access to the same information.

Each object can exist in four different states, partially based in [14]:

1. Under Implementation: The object is being built and/or developed, and it is not yet used by other objects belonging to the structure.
2. Productive: The object is an active part of the structure, and interacts with other objects.
3. Decommissioned: The object still exists, however, it no longer plays a significant role in the structure, remaining in a "zombie" like state.

4. Deprecated: The object no longer exists in the real world structure, and information about it is maintained for historic purposes.

View then manifest theses states through the use of colors, with each state having an identifying color (Ex: red means deprecated). Initially, upon presenting a view to the user, the platform does not distinguish between these state, however, on top of the view, a time slider is positioned.

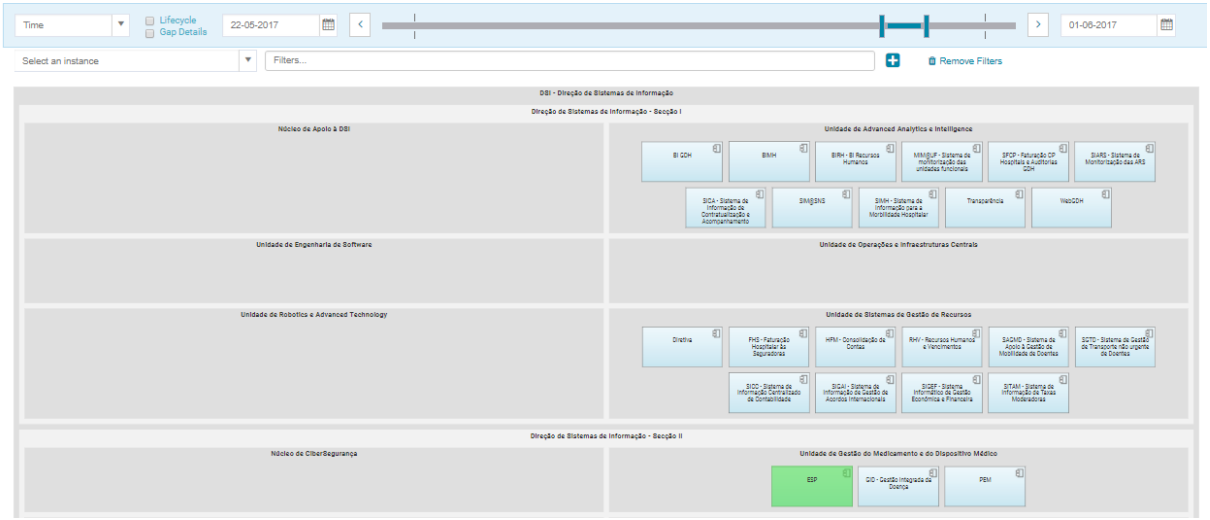

Figure 3.28: Object with highlighted green color meaning that a specific object, within the time span defined in the time slider above, become productive.

This feature allows users to analyze how time affects the structure of the organization, or a specific viewpoint, with having to manually retrieve information about important dates,providing this information in a more direct manner than before.This feature depends only on the information inputed in the time-related properties.

This allows for more informed planning decisions, as well as using this platform to to plan future changes or work schedules. For example, if a team is responsible for a solution that will now provide application services to other solutions, it is possible to know at what dates should these changes be made, what Applications, Nodes, Actors or System Software will be affected, and take the appropriate measures.

3.5.1 Project Class

As mentioned before, a concept that represents a set of transformative initiatives exists, in the form of the ”Project” class, already introduced in section 3.1.14. This class was created by the necessity to support the representation of planning within the Atlas tool.

Even if the dates are correctly used in every object, the users still had to individually check the changes planned for the objects that belonged to the same project, since in the context of SPMS and eSIS, changes to the existing structure come in the for of projects, that have schedules and a set of objects that are affected, in different ways, by it. In order to group this set of influences and the object upon which they happen, this class was created. Each object of this class should represent a project,

along with the objects that are affected by it, and in what way they affect, since a project represents the force that cause a change of state to some object, that happens at a certain date.

Organizing this information into a single class was made so that planning could become even easier, by aggregating all the components of a project into a single class (including the Actors with responsibilities towards it, such as the Project Manager), the users that wish to be informed about a certain initiative happening in the organization, need only to consult this class or views built around it, such as the ”Project Dependencies”, that shows what Solutions are involved in a certain project, as well as the application that compose them (influences the project has over other elements such as actors or Nodes were not yet described, by decision of the SPMS architectural team, that wishes to follow a more iterative approach to this information, meaning that at the time of writing, influence over Application class objects was the main worry, but mechanisms for representation of the influence over other classes of objects exist, but are not used.)

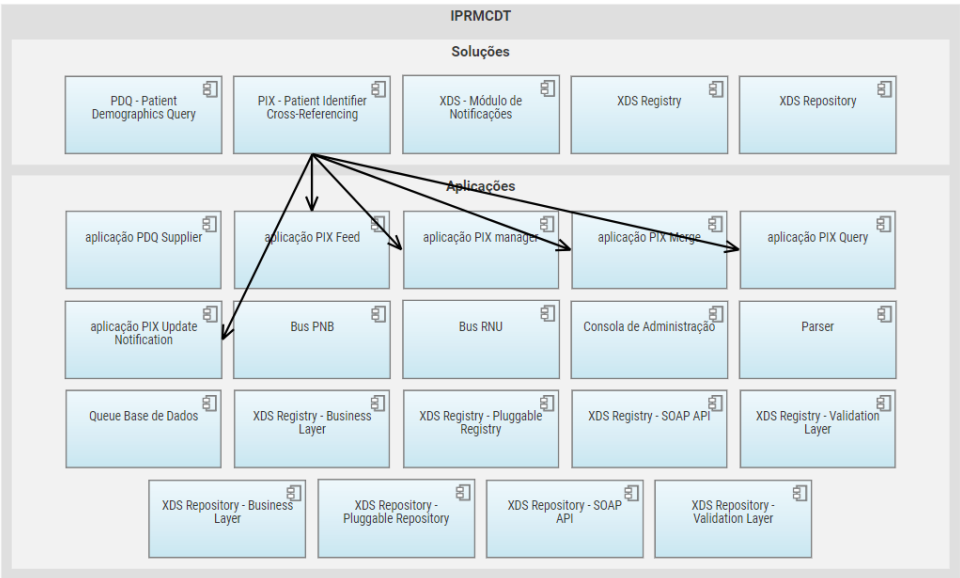

Figure 3.29: Project Dependencies. This view exposes the Solutions that are influenced by the argument project, and the application that they aggregate.

Projects can of course be divided into several sub-projects, and such information is of course also possible to visualize, with the view ”Project Structure”

3.5.2 SPMS Project Management Platform

Through the development of the architecture project, SPMS coordination requested that a project management platform (PMP) was created. This platform would take the form of web-based and mobile application that would allow user to be informed of all the dates relevant to projects, Actors responsible for the management of the project and the coordination areas these actors and projects belonged to. At first glance, this would be an initiative that was out of the context of the architectural project. However, the architectural team and the author came to the conclusion that the information that existed in the ATLAS repository could very well be used to feed this new platform. Relations between classes and their

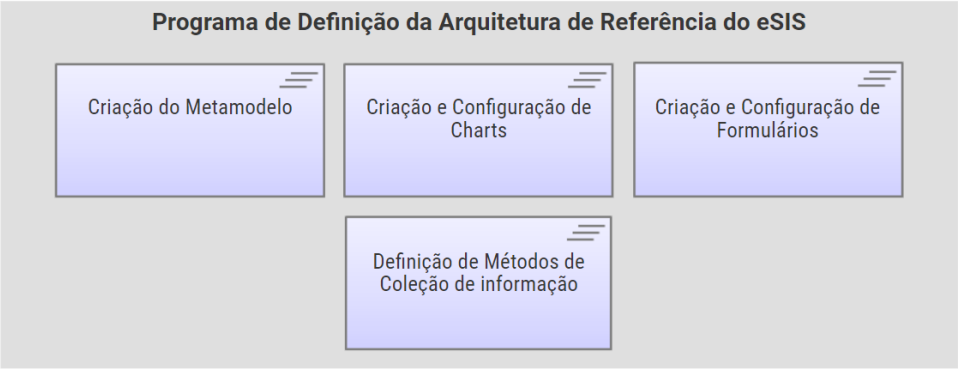

Figure 3.30: Project Structure. This view exposes the internal sub-divisions of an argument project.

properties could be used to clearly define how information should be organized, and this organization in turn would provide the necessary information to structure and ”feed” this platform.

In order to accommodate the requirements for the PMP, we organized the information as presented in Fig 3.28, following the real world hierarchy that exists within SPMS

In this structure, we have a direction area, that in turn has an Actor responsible for it. This direction area aggregates a set of smaller departments called coordination areas. Each of these coordination areas has an Actor responsible for it, the coordinator. These departments are responsible for coordinating the projects, with one of the members of it assuming the role of project manager. Projects are then associated to a set of objects that they influence.

By inputting the information in such way that it respects this defined structure, the ATLAS repository could serve two purposes: feeding the ATLAS tool and also the in-house, proprietary SPMS PMP. The architectural team become responsible for creating the forms and classes necessary for this information to exist in the repository, and also inquire organization members to complete this information. This feature allows the organization to have a more flexible solution to the planning of projects and changes to the structure, while at the same time providing support to the management of these issues, since information relevant to a large number of individuals is now accessible to them in a centralized platform that follows the same rules as the established architectural meta-model.

The development of the tool was of the entire responsibility of the SPMS development teams. The architecture team’s sole purpose was to facilitate and enable the creation of information to feed it.

3.6 Use Enterprise Architecture to solve Operational Issues

In section 1.2.3, we explored how SPMS struggles to find a way to use architecture to solve operational problems, meaning the organization could not find a way to use EA as a means to perform tasks that where necessary to complete the work-flow. It was up to the architecture team to figure out this method.

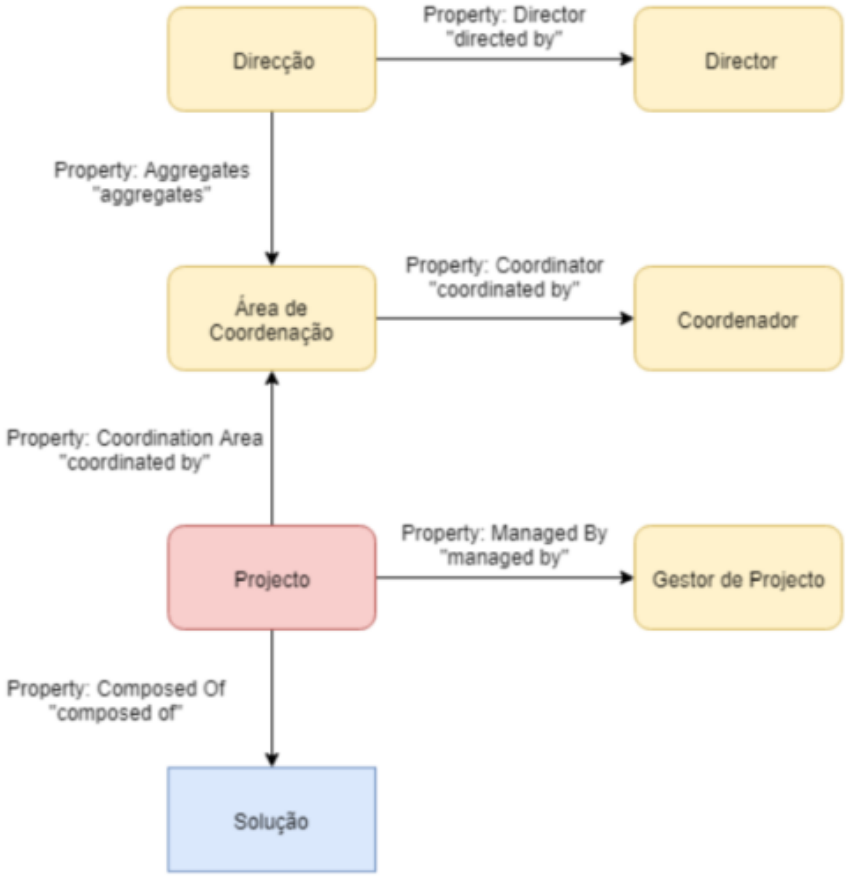

Figure 3.31: Hierarchical structure of Actors and Projects. This figure represents how the information for the PMP is organized.

3.6.1 General Data Protection Regulation (GDPR)

During the development of the project, the necessity to conform to the newly introduced European Union data protection regulation came about. Public and privates entities all around europe were forced to be ready for auditing operations that would validate if the organization was respecting the integrity of the data used in their businesses (and even non-profit organizations were under this regulation). SPMS and the eSIS ecosystem work on a daily basis with large amounts of personal information from different people from all over the country, and some foreign, with some of this information being classified as ”very sensitive” (blood types, clinical results etc.). The architecture team saw in this situation an opportunity to use EA and the ATLAS tool to satisfy, while partially, some of the requirements imposed by GDPR. The thought behind this idea was not to completely satisfy GDPR auditing requirements, but find a way to support the organizations in this aspect, since ATLAS representation does not hold legal value when presented to the audit authorities, however, they can help the organization understand their internal access rights, permissions and influence each actor has on certain information.

This would also help members of the organization effectively perform operational tasks, since the collection of information to correctly answer auditing evaluations was a task that the organization had

to perform by law. While the view and information deposited in the ATLAS repository regarding data protection concepts would not automatically be transformed into legal documents, it could help members of the organization have more direct access to it, and then create the respective legal documents.

Given this situation, the main worry SPMS and the eSIS entities had was to understand what actors could do to/with information that existed in the solutions they were able to use, as well as define who could use what solutions. In order to tackle this issue, the architectural team developed a set of relations between classes that would help create views and a structure to accurately represent this information. This model is presented in Fig 3.29

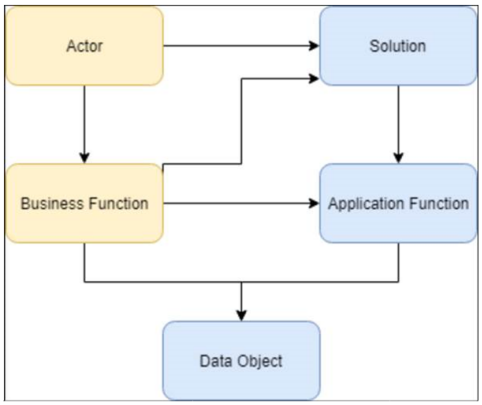

Figure 3.32: Logical structure behind the mapping of GDPR concepts within the Atlas tool.

The idea behind this structure is that each Solution has a set of profiles (Actors) that represent roles of people that interact, to varying degrees, with it within itself (Doctor, Nurse, Developer etc.). Each Solution had a set of Application Functions it could perform (as described in section 3.1.6) and these Application Functions had a set of influences over the information (Data Objects) that followed a Create/Read/Update/Delete (CRUD) paradigm. Profiles also had a set of actions (Business Functions) that they could perform and each of them would only be possible if an Application Function within the Solution existed that supported its existence. The profile actions would also have an influence over information, following the same paradigm as Application Functions. However, context was missing, meaning that, for example, while we knew a doctor could access a certain information, could he do it in every Solution? To overcome this, we added a relation between the Business Function and the Solutions (Application class object), meaning each Solution of Profiles, Application and Business Functions. This way we were able to create context for the profiles, meaning that even if that profile has the permission to permission to perform a certain Business Function, that same Function must be recognized by the Solution and support by one or more of its Application Functions. This however, requires the premise that every object representing a Business Function is unique, wish is true in almost all of the Solution. However, the problem of duplicate Business Functions is a reality, and in order to overcome this, since it is a problem with a very small occurrence (according to the members of the organizations), if two Business Function are equal but exist in different Solution, we name them in such way that it is possible to distinguish the two. The problem of excessive duplicates is not, once again according to the organizations, a considerable problem.

With this structure, we could then represent access and permission relations using views, such as the ”Profiles By Solution”(Fig 3.30) and ”CRUD Matrix” (Fig 3.9)

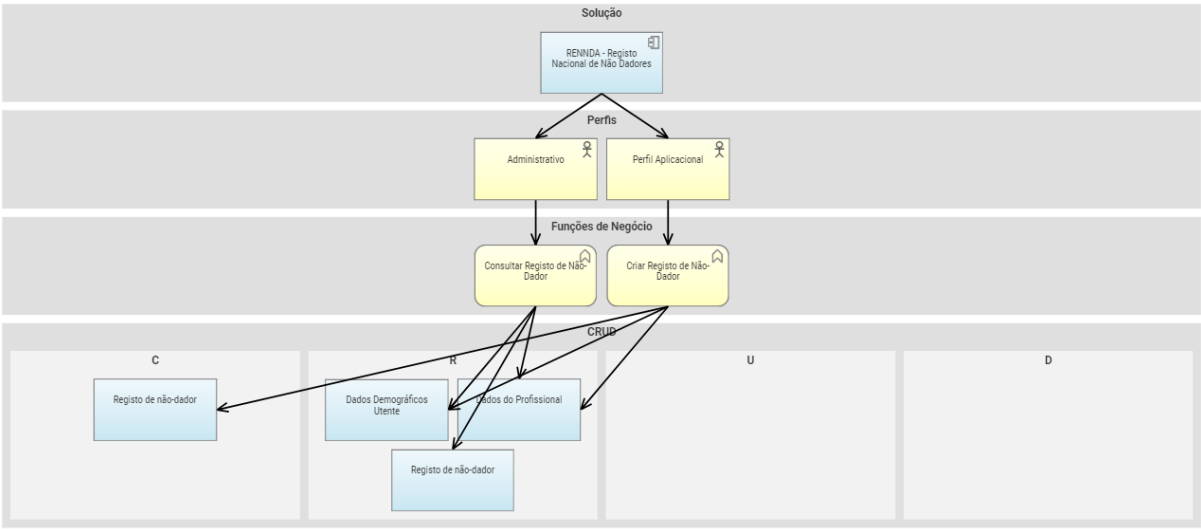

Figure 3.33: Profiles By Solution view. This view follows the exact same logic as the CRUD Matrix view, but in the context of Business Functions.

Using these resources, the organizations can now have a clearer idea of the permissions associated with the Actors, Solutions and information are structured. The collection of information for GDPR purposes can be partly performed resorting to the ATLAS, thus providing a way to use EA as a means to perform these (specific) operational tasks.

3.6.2 Project and Life-cycle Planning/Scheduling

Following the same approach and logic used in the GDPR problem, we can use EA as a means to perform other operational task, namely, if the task at hand is planning. For someone responsible for the scheduling and planning of changes to the structure, creating information regarding those aspects is in fact an operational task. Using the already explored planning features available in the ATLAS tool (section 3.5), the person responsible for the planning of these projects or changes to the objects can use the ATLAS tool to save these changes, making them available to everyone involved, instead of having to later distribute the information through emails that can easily be forgotten.

3.7 Document relevant points of the Project

We mentioned in the Introduction section that SPMS was, before the beginning of this project, an organization that had no previous experience with Enterprise Architecture projects, meaning that guidelines and solutions to common problems that happen during these types of projects were not well known within the organization.

Since we could not solve this problem before the actual start of the project, and during its development we could only mitigate it, the architectural team chose instead to gradually and continuously

build a set of documentation and supporting material to register this type of information, based on the functioning of the project. This was a solution that was more oriented to the future, since many of those working in this project may no longer be in the future, and with approval from the organization's management, registering certain aspects of the project would provide long term benefit, should the organization embark in a new architectural endeavour or continue this one with a new team working on it.

3.7.1 User Manual for ATLAS

Since this project is highly dependent on the tool being used to present and store the information retrieved during the endeavour, creating a "taylor-made" manual for ATLAS seemed like an absolute necessity. This manual would focus on exposing the basics of utilization of the tool, such has describing the menus, explaining what symbols and colors mean in the context of the tool and basic configurations methods. Part of this piece of documentation was built with the common user in kind, people without any technical knowledge of the inner workings of the tool, nor the need for such. In order to built the most efficient and rich user manual, we interviewed some users to better understand their main difficulties and worries when using the tool. These interviews were used not only for the writing of the manual, but also as sort of product evaluation, meant to point out possible improvements both in functioning and presentation.

The manual also includes a more technical part, directed to those responsible for managing and configuration the tool, with it's use meant to be internal to the Link Consulting firm. This part documents frequently Asked Questions (FAQ) on how to solve and perform some of the client's most requested tasks (based on this project), as well as integration, information importation and managing issues.

3.7.2 Weekly Development Reports

In order to maintain historic information of the project's development and document everything that happened during it, including bugs within the platform, user requests and maintenance problems, at the end of every week during author's participation in the project, a summary report of the events that took place during that week was written. This helped keep track of various aspects of the project, while at the same time serving as a tool of communication between the author and the project managers at Link Consulting (since the field work required physical distance, the author spent most of the time in SPMS headquarters, not at Link Consulting).

3.7.3 Meta-model Description

Since one of the pillars of this project is the introduction of a brand new language in the form of a meta-model, documents explaining it are a priority. As such, the architectural team decided to create a document that explained in full detail the meta-model, while at the same time now requiring a full reading to get a grasp of some of the basic concepts.

Each of the classes existent in this meta-model are explained, as well as the relations they establish between other classes, along with the concepts in the real world they try to map. It was done in a way

that explained everything to the up most detail, should the reader wish to fully understand the concepts. A superficial reading should also allow user to understand the basics.

This document was shared with all participating entities, and in most cases, served as the only way to communicate the meta-model, since presentations in every single entity were not possible due to the time and monetary costs associated with that option.

3.7.4 Report of Progress Problems

The final piece of planned documentation is document where all the main problems and difficulties of the project are documented. By ”main problems” we mean problematic situations that happened during the development of the project that represented considerable roadblocks to the it’s development, be it technical difficulties regarding the deployment of the ATLAS tool into SPMS servers, or more functional problems, like a list of struggles members of the organization had. The findings documented in this document will be further explored in the Results section (4).

3.8 Use of Charts, Information Structuring Rules and Pilot testing to facilitate multirepository repository management

Perhaps the most difficult challenge of all the implementation process, the management of a large numbers of repositories and the aggregation of all their information into a single, centralized repository was the most demanding task of all the project, and it’s ultimate goal.

As previously mentioned, the architecture project that is the subject of this thesis, does not limit itself, in terms of information, to the SPMS organization, since it also includes the information ecosystem known as eSIS. This ecosystem includes around 66+ repositories that represent public health entities. Each entity has it’s own specific context, meaning Actors, Information Systems and Infrastructure exist in such way in that they make sense in the that context, and does not necessarily follow a template shared among these entities on how things are organized, although the variations between them are not so big that it becomes impossible to find a common thread. The challenge was finding a way to extract information from multiple sources, describing multiple entities and find methods and tools that would help us coordinate the information, to make proper use of it.

3.8.1 First Steps

Due to the size of the task at hand, the tackling of this challenge had to be planned with considerable care in order to maintain the order between repositories. The first decision to attain this goal was to use the same meta-model to describe the internal structure of each entity. This was done since if we plan on aggregating all the information into a single unified repository, all information had to be subject to the same rules. To guarantee this, every repository representing an eSIS entity was forced to follow the meta-model defined for the SPMS repository. The team found that the level of detail inside each entity

was not meant to be as deep as the one existing in SPMS, but not so generalist that the information provided by the entities became useless. However, the structure of entities does not follow the same pattern as SPMS. In order to overcome this, we defined that each repository must have an object of the Actor class that represents the entity, and each object that existed in that repository needed to have relation between them, so that from any object, we could reach that ”main” actor. This means that if, for example, a random system software was picked, there had to exist a relation between it and the application it serves, a relation between the application and the solution it was aggregated by, and from that solution a relation to the department (if applicable) that is responsible for it, and so on until we could reach the entity Actor. This rule served as the basis for the structure of the information of the repositories.

The architectural team also decided that most views created for the SPMS repository made sense in the context of the entities repositories, such as the ”solution Organic”, with slight changes made to their configuration in order to accommodate specific cases regarding how information was structured, such the previously mentioned ”main entity” Actor. Others where specifically created for these repositories. One example is the ”Provider Organic”. Many of these entities, due to their nature, do not develop their own Solutions, and instead depend on third-party solutions. These solutions are provided by external actors we denominate as ”Providers”, which are represented as Actor class objects, that in the property ”Type” are identified in that way.

Figure 3.34: Provider Organic View (Unidade de Saúde Local de Matosinhos). This view presents all the external Actors that provide Solutions to the entity, along with the Solutions themselves.

These decisions represent the first steps regarding how the information for the repositories would be structured. This allowed the architectural team to create a set of rules that all information from the repositories had to respect. The next step was to educate the architecture managers from the entities on the meta-model and how the information should be structured.

3.8.2 Meta-model and Information Training for Entities

While members of the entities were not suppose to have a very deep understanding of the meta-model, it was still necessary that on the entity side of things that someone had some understanding, in order to not only to help those in the entity and serve as a means of communication, but also to be responsible

for the quality of the information that is inputted in the repository. As already mentioned, being present in every single entity and explain the whole logistics behind the whole project was not a feasible alternative due to the costs associated.

In order to overcome this roadblock, the team decided to take a iterative approach to the repositories, meaning that instead of going through a distance-call meeting spree with all the 66+ entities and then allow them to start introducing information, the team choose another alternative. The team debated and came to the conclusion that the mentioned method would be extremely inefficient, since meetings would be too extensive and exhausting for team, which could compromise their quality and hurt the correct understanding of the work being performed by the project. The team also feared that if all entities were to be allowed access to the platform all at once, too many problems and questions would ensue, most of which the team would not be able to properly tackle.

Instead, we opted for a different approach. Initially, ten entities (chosen by the interest they had independently presented in the project) we chosen to serve as the subjects for the pilot testing. The idea was to have a concentrated group that would serve as a demo for what would be done on the rest of the repositories. This smaller scale approach would allow the architectural team to work in a more focused manner, better understand the struggles of each entity and help shape a new way of teaching the meta-model and present the project. After this step, and using the feedback from this test group, we would then select another set of ten entities and perform the same tasks, hopefully with more efficient methods.

The test groups, as already mentioned, were formed of people representing ten different entities, with the number of actual individuals varying from a single representative to a group no larger than three people. The people representing the entities were chosen by the entity itself, with most of them being connected to the IT or Infrastructure departments, something that made the architectural team's job easier, since there was some background in at least some of the concepts used. The first step in this training was to perform a distance-call (using software such as Skype) to formally introduce the project to the representatives. The EA team would go through the meta-model, explaining the basics, such as what each class should represent, rules of how information should be inputted, methods for inputting information (XML/Excel imports, forms etc.), the benefits of the project and using the ATLAS tool on a regular basis. Many of these meetings would focus more not on the technical aspect of the project, but a more ”political” aspect. While many of the participants saw the benefits, those with a more ”conservative” mindset were more difficult to convince.

After the meeting, the representatives were given three to five days to consult the documentation (sent a day before to the entities' representatives), so that they could find aspects they were less comfortable with or any doubts that were not fully explored during the meeting. Following this, the architectural team would then give repository access to that entity, so that information could start being inputted in the repository. While the team did explain the meta-model, using practical examples, such as the richer SPMS repository, we were fully aware that mistakes and errors in the information would still be inevitable, and that was taken into account.

The original idea was to repeat this cycle until all the entities would be properly informed and with

enough information inputed so that at least the ”Solution Organic” map would have a considerable amount of information presented in it. However, there were some setbacks in this plan, due to scheduling problems with the entities and some cases where the entities were not available for a meeting.

3.8.3 Importing Information from Multiple Repositories: The Batch Feature

The goal of this part of the project was to populate the entity repositories with enough information that could then be transferred to a central one, named ”eSIS”. However, the traditional methods of information input made sense in the context of a single repository, not a multi-repository scale. The traditional ”export then import” would not be a feasible method, since it would imply manually exporting information from each class from all repositories and then aggregate in the eSIS repository. Not only that, but maintaining information updated would also present another problem. If the team wished to make the multi-repository paradigm a reality, another method had to be created.

With the help of Link’s ATLAS development team, a new feature was conceived to overcome this issue. A Batch is a tool that allows the user to export information from various sources and select the repository where the exported information is to be imported. Batches are composed by a set of ”Jobs”, actions that are executed in the order defined by the user. Each Batch can be configured to serve the needs of the user, given only that the scripts used and the input file (that contains the necessary arguments) have the necessary information.

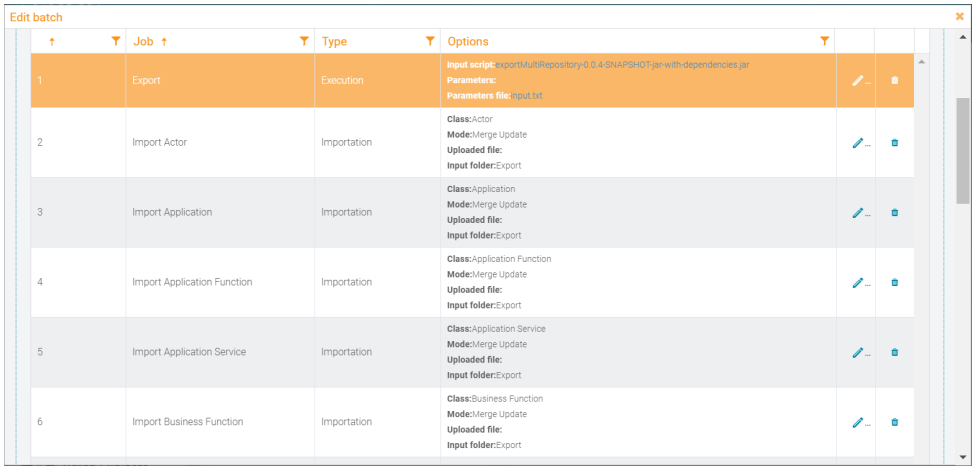

Figure 3.35: Batch configuration menu. This job contains a set of Jobs that are executed in the order they are presented. First the information is exported from all the repositories defined in the ”input.txt” file, using the .JAR script. The information is then stored in a folder, from which the following jobs take the information.

Through this feature, the architectural team found a way to take information from a large number of repositories, and aggregate it into a single point. However, this feature does have it’s limitations, which required the team to come up with a set of rules to ensure that the information stays consistent.

Batches import information from the repositories and they are ordered in the input file, meaning that if the field ”REPOSITORIES” in the input file is written like this: ” ’A’,’B’, the respective Job will first export and import the information from repository A, and only after completing that step will it start exporting

information from repository B. When the information is put together, the ATLAS tool joins any instances of a class that have the same exact ”Name”, meaning that even the slightest difference (it is case sensitive) will result in too different instances, even if it only meant for a single one to exist. When two instances of the same class have the same name, the ATLAS tool joins the two into a single instance, that retains the properties of both. However, if the property is of the ”Text” data-type (classification for the properties type), ATLAS will only save the latest value. Using the same ”A” and ”B” example from above, if the order is ”’A’,’B’”, and the same object ”X” exists in both repositories, with the property ”Description” ”Hello” in repository A and ”World” is B, in the eSIS repository only ”World” will appear. This can become a problem, when different repository users decide to input different textual values in these properties, because the last one to be processed will essentially define the final value, which may not be accurate.

The problem of incorrect information is also a big concern in this process, since even though we may be able to join all the information into a single repository, that does not necessarily (and most probably will not) translate into ”valid” information, meaning, structural mistakes will happen, such as Applications instances that represent solutions being aggregated by other instances (violating the rules set for the meta-model), so having some information cleansing mechanisms was absolutely necessary.

To tackle these problems, the team came up with a logical structure for the eSIS repository, which would serve to solve the textual properties problem, and try to mitigate the information quality control. Before the changes to the structure, the way the flow of information was organized was as described in Fig 3.33, with repositories having their information extracted in the order they appeared in the input file, using the Batch feature, and all information would directly go to the eSIS repository.

Figure 3.36: Old eSIS repository flow of information structure. All entity repositories have their information extracted by the arbitrary order defined in the input file.

Upon rearranging the structure, the flow of information would now work according to what is presented in Fig 3.34.

The flow of information now established would work by first extracting information from all the repositories, regardless of order. However, after those repositories have their information exported, the last one would be the ”eSIS Master” repository. This repository serves the purpose of registering the absolute value for the textual properties, meaning that in this repository the architectural team would write the values in textual properties that corresponded to the ”absolute truth”, such as descriptions for the Solution and Applications developed by SPMS, the Solutions and System Software developed and provided by third-parties. This way, we would be able to guarantee that, despite what the values inputted by the

entities, the correct ones would end up remaining at the end of the export and import of information. This of course is solely applied to the textual properties, as relations between instances described within the repositories must be allowed, so that entities could describe their internal structure, even if that served as a gateway for more mistakes to happen. This compromise was necessary, since restricting what relations can exist is not only extremely difficult in terms of implementation, since ATLAS has no awareness of context and simply shows back what the user defined, but would make the whole entity's personal repository redundant.

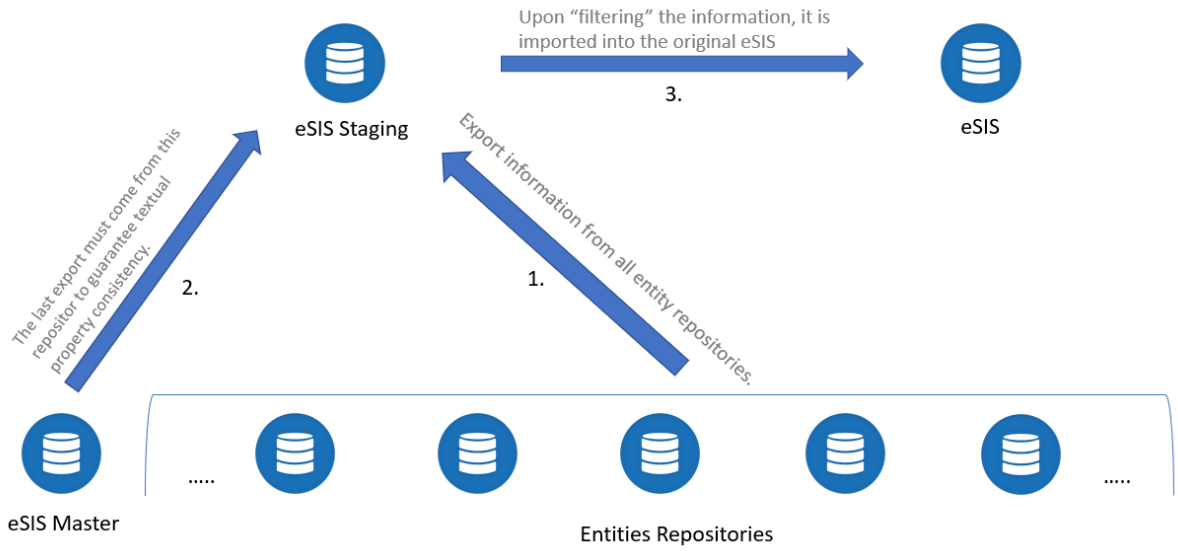

Figure 3.37: Current eSIS repositories flow of information.

Unlike the previous structure, the information, upon being completely exported, including taking the textual properties information from the "eSIS Master" repository is not directly imported into the "main" eSIS repository, and instead goes to "eSIS Staging". This repository serves as a "filtering" stage for the information before being imported to the main eSIS repository. As already mentioned, collecting information from various sources brings about a lot of mistakes regarding that same information, and exposing that information directly to users that wish to consult the eSIS repository would be inefficient, since much of it still needed to be treated and corrected, and could lead the user to take the wrong conclusions. In this repository, treatment of information occurs, such as relations that violate the meta-model structure are corrected, duplicates are eliminated or merged with the correct instance, and some may even be renamed.This of course is not a perfect system, since it still requires information to be evaluated by members of the architectural team, and the amount of time and understanding required is more extensive that one would wish for. In order to avoid this inefficient consumption of time, a the team implemented a policy regarding information that stretches too far from the set of rules or simply has no context or way to understand. In case the above criteria are met, meaning there is no way for the architectural team to understand the original intention of the entity or what was trying to be represented, that information would be erased from the "eSIS Staging" repository. Instances of the Application class, for example, that represented Application Releases or versions (from the physical part of the meta-

model) and were not correctly associated with the original Application, for example, would be erased.

The logic behind this policy is that treating information can require too deep of an understanding of the entity's structure, something that members of the architectural team do not have, so instead of losing large amounts of time just understanding a particular instance, it would just be better to erase it from the repository, and notify the entity of their mistake, and possible solutions so that it can accurately describe it's reality. Also, maintaining wrong information or information that the eSIS repository managers did not understand would only serve to further confuse the users if imported into the main eSIS, and create more unnecessary complexity within the repository. By eliminating this inconclusive information, we are shielding end users from misinterpretation and information that would only take space and be counted in the construction of charts of views, while not really making any sense within that context.

Of all the steps in this multi-repository information flow, information quality control is by far the most time consuming, but a necessary endeavour to guarantee accurate and consistent information to the users. By setting up the structure described above, information control becomes a easier task, and the amount of extra steps implemented allow for more clean end information.

After the information cleansing is completed, it is them exported and imported, also through a Batch, to the final and main eSIS repository, were users can use to generate maps, charts or queries.

3.8.4 Charts as an Information Quality Control Tool

Charts have already been mentioned in section 3.4.1, however, the full extent of their usefulness in the implementation has not yet been explored. Charts present statistical meta-data regarding the information that exists within a repository, as such, they also exist in the eSIS repositories and have their purpose there. In the eSIS Staging repository, charts play a particular role. In order to further improve the efficiency of the quality control that occurs within this repository.

To help this cause, charts such as "Number of Solutions without Provider" allowed users and architecture teal members understand which information was incomplete, as all Solutions should have associated to them the actor responsible for providing it. In the same vein, charts like "Node without Manufacturer" or "Applications without Systems Software" allowed users to count how many instances were lacking vital information. For charts like these, the objective is that present the smallest number possible, which translates in information that, while may not be 100% accurate, the properties necessary are all filled.

While counting the number of times same type of mistake exists in the repository is an improvement over not having mechanisms like this at all, it is still relevant for the users to know exactly what instances fit that criteria. As such, the user can use the queries used to generate the charts to find those specific objects, using the query explorer feature.

This feature improves the treatment of information by allowing the users, be them regular users or architecture team members, to pinpoint the problems in the information and the instances that suffer from it. May the need arise to find a particular issue with information, a new chart of query can be configured to satisfy those needs.

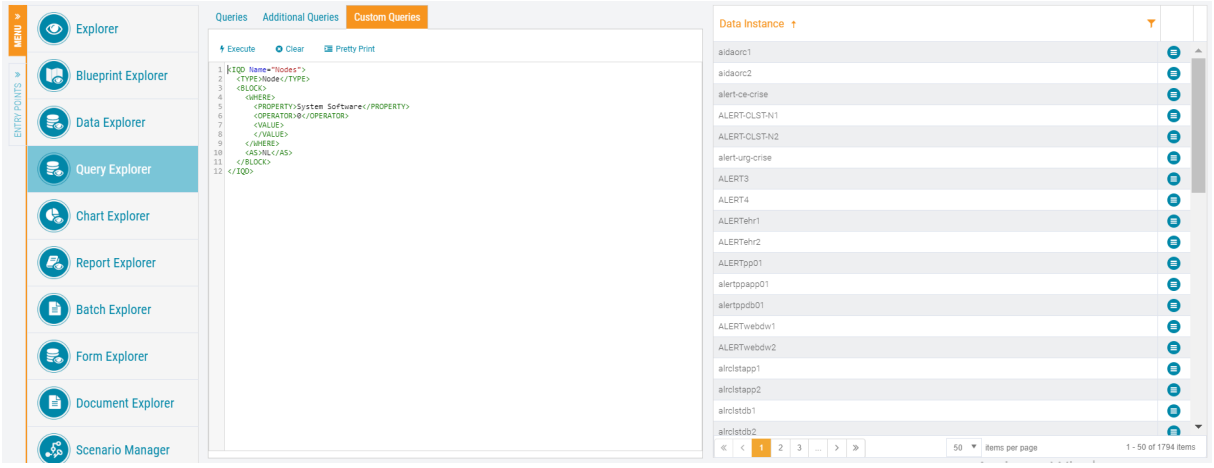

Figure 3.38: Using the query used to generate the "Nodes Without System Software" Chart in the Query Explorer, users can identify the specific instances of the class Node to which the criteria applies.

3.8.5 Scenario Manager: Controlled Change

Controlling the quality of information coming from various sources becomes a difficult task due to the regular (and somewhat expected) amount of errors in the information that users and managers from different repositories input into the platform, and in order to mitigate this issue, various already mentioned mechanisms are used. However, most of the present solutions that were implemented focus on the information after it as already been inputted, meaning the treatment begins after the mistakes are already done.

While not a completely error-proof method, prevention does help mitigate some mistakes before they happen, which led the architectural team to use the Scenario Manager feature from the ATLAS features. This mechanism serves as a way for the managers of each entity to check the changes that other users proposed , allowing for a review of the changes before they happen, guaranteeing, to a degree, that information has the expected quality before it becomes a part of the architecture.

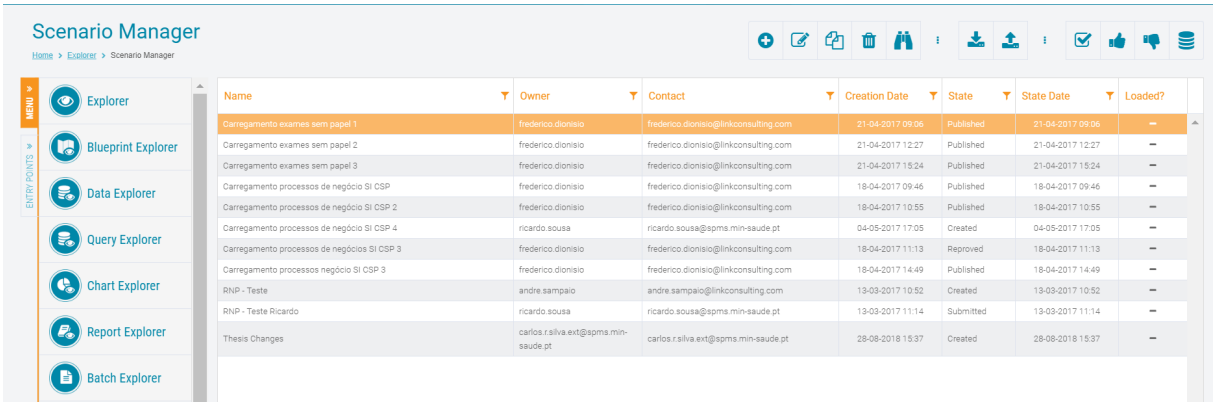

Figure 3.39: Scenario Manager main screen. Allows the user to visualize all existing scenarios and their respective state. Authorized users can also use this screen to validate or deny the proposed changes.

The Scenario Manager works by following a hierarchy of responsibilities. Users with permissions to edit the information can create their own scenarios, which consist on proposing changes to both existing

information, be it creating new relations or simply alter the values of some given property, or adding or erasing information. These changes are then submitted for approval. The approval of the changes proposed by a scenario are the responsibility of the repository manager. The manager must evaluate the quality of the alterations before putting them in the structure of the repository. First, the changes must be validated. Validation consists on running a set of queries that make sure if the information and changes proposed don't create inconsistencies in the information already in use in the repository, such as duplicate instances, check if Solutions are not aggregated by any other application instance etc. If the the queries return an empty set, that means that the changes proposed do not create any inconsistency in the information and respect the established structure, meaning they can go through the next step: Approval.

If the changes proposed do not create inconsistencies in the structure of the information, that does not necessarily mean that the information is "correct", as in, accurately represents the changes that will happen in the real world or even properly represents the reality. Since ATLAS is a software, and as such, it is not aware of the context of how the real world is, only receiving information that is structured according to the rules set by the architectural team, human evaluation of the validity of the information must occur. An example of this may be the implementation of a new Solution. The user that proposed the description of the new Solution proposed an internal structure for the Solution that the manager does not agree with, and therefore, the manager can disapprove of the changes. The user can then check the status of his/her proposal, and in case they are not accepted, they can redo the proposal or simply abandon it.

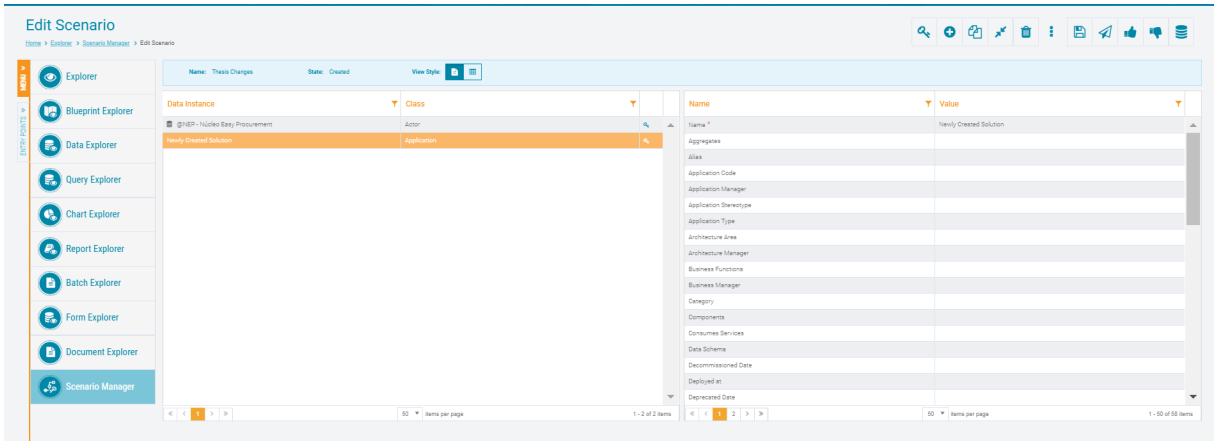

Figure 3.40: Edit Scenario Screen. The user can select instances he/she wishes to change or create, from scratch or using information that already exists in the repository. Upon configuring the changes, the user must save the scenario and then submit it for the approval of the manager.

If the scenario is approved, the manager can then publish it when he/she chooses to do so. Publishing a scenario makes the changes proposed by it effectively part of the information, meaning that the information in the repository also includes the changes made in the scenario.

This mechanism represents a more careful and approach to the input of information in the local repositories. Instead of having users directly inputting information within the repository that effectively stays there, but may not be valid or even correct, the Scenario Manager allows for a user (the manager)

that is selected within the entity as the most capable in terms of understanding the concepts behind the EA project, to control what information is inserted in the repository, making the process of joining all information in the central eSIS repository an easier task, since (in theory) information that goes through such process is ”cleaner” and less prone to mistakes, such as the incorrect naming of popular solutions among entities (for example, incorrect use of caps lock in the name of the instance).

Chapter 4

Results

In this chapter of the thesis, we will expose the results of the solutions implemented, describe in detail the responses and feedback received from these changes, as well as personal insight. All this will be evaluated according to the defined success metrics, in order to compare the goals defined by the organization and the author against those initially projected.

4.1 Results of Implementation : Stakeholders

As previously described, each part of the proposed solution had the objective of tackling a specific part of the ”problem”, in an approach that was more focused on covering the breadth of the problems the organization faced, instead of their depth.

Some parts of the solution where easier to implement than others. The easiest ones were the mostly technical, regarding the configuration and creation of systems within the Atlas platform to support our project, in other words, parts that did not require direct input of the organization to be performed. These include the batch features, the configuration of maps (created according to the needs of the organization) and creation of classes within the system. with the support of the talented Atlas development team, these endeavours were easily overcame.

The greatest challenge, however, was the political aspect connected to these topics. Due to the nature of the project (consulting) and the breadth tackled, many different agendas and different personal opinions contributed negatively to the development of the program, specially one that required the construction of a new culture from the ground-up within an already well established environment.

This was specially notable in the definition of the meta-model. While some of the most basic concepts where defined by the project team, in order to guaranty that a basis for building anything else existed, such as the division of the application class in a three level hierarchy (Solution, Application, Application Component), much debate arose from what belonged in each of the levels, since different teams had different ideas of how systems worked, and how they should be mapped. This situation verified itself with almost every class created in the meta-model, with classes like ”Actor”, ”Project” and ”Location” (for representing the most straight forward concepts in the eyes of the organization’s members) being

the only exceptions. This conversations with members of the organization regarding the meta-model revealed that one of the reasons for this amount of discordance when defining a new class was due to the low EA awareness of members of the organization. This situation is similar to the one found in [40], where the author verified that the lack of overall EA awareness hindered the development of the project.

Unlike the situation this author describes however, the top management of the organization where actually the ones, within the organization, most aware of the concepts of Enterprise Architecture. This showed to have helped the project in a fundamental way, since having individuals in higher places within the organization's hierarchy allowed some of these roadblocks to be more easily mitigated, since some of the concepts and opinions could be transmitted directly toward management position who in turn would spread the message throughout the teams, and these would be much more receptive to them since it came from "inside" the organization. This would manifest itself in times where there was a lot of debate of how a system should be mapped, the architectural team would have a hard time guiding team members to the most appropriate solution. When this happened, the team would reach the management positions responsible for the EA project to try and explain and convince them of our positions (most of the times successfully), and in turn the management would approach the organization members with the same solution, and they would be much more receptive of the message.

This dynamic took a roughly 2-3 months to establish, and once it was set, decision making processes within the project ran much more smoothly, which helped the project develop faster.

This specific theme was to be expected due to the fact that the worked performed for this thesis was within the context of a project being delivered to a client. The author verified that many of the characteristics described in [9] could be verified first hand: the Project was directly associated with organizational processes (business processes the people within the organization performed) and systems (the project's main focus was mapping the existing IT infrastructure and connect it to the business processes and actors); Defining the Scope showed to be a hard task, due to the uncertainty of those who provided us the information, and due to the fact that most of the elements mapped were "uncharted territory" (no previous reliable mapping had been accurately performed, so most systems were initially black-boxes for the EA team); Due to the size of the organization and the previously described political struggles, the Change control was a very demanding task, as almost everyday new requirements would be presented, and maintaining all of them coordinated and coherent took a lot of effort and documentation. Also, the fact that the Speed of these changes was considerable did not help the project's smoothness; The Evaluation Indicators was also to track down, due to their subtlety (implementing a new property to a class or a brand new class would only produce feedback from users same time after they were implemented).

We also came in contact with factors described in [10] as the main reasons for failure in these types of projects: the overall lack of EA awareness hurt the development of the project. This, however, was a premise the we conted with, since this organization had no previous experience with EA project. Nonetheless, having the extra step of having to explain fundamental EA concepts slowed the project, which would develop much faster if the members of the organization had previous experience.

"Limited commitment from interested parties inside the organization" and the time it takes to set up an architecture were the two failure factors (listed by the author) that we faced more directly. The limited

interested become apparent in situations such as the effort from people of the organization, which due to them having other justifiable priorities, did not match the effort needed to make the project advance at a desirable pace. Resistance to change (as already discussed) also hurt the project, as many people did not see value in the proposition of having an enterprise architecture established to support their processes, with the common ”Why do we need this?” statement. To counter this type of perspective, we decided to use the results of information collecting and representation, which lead us to the second major failure factor we encountered, the time. Enterprise Architecture are complex to create and take a lot of effort to become well defined. While our projects was not the creation of an EA, we had to map one from zero. This established a pressure point on the project: We needed to present results fast, and for that we needed the feedback and information from the members of the organization, but to convince them that their effort would payback, we needed proof, whcih lead us to step one. To counter this situation, the EA team made a strategic choice: instead of asking every team to provide us the necessary information, we would instead start ”small”, meaning a team was chosen for the pilot. This pilot would be described mapped to with rigorous detail, so that every application, actor, software tec, would be taken into account (just for this case). We would then use these well mapped systems to use as an ”argument” in favour of our project.

Upon trying this new approach, people felt more comfortable and reassured that this project would benefits them, since they would look at the capabilities of visualization of information using the EA tool, applied to other system that was not theirs and would want that in their context. By convincing people, the effort needed to map the architecture was spread across multiple individuals and not only the EA team. However, the EA team would always consult and give their advice on how things should be mapped according to the rules defined.

4.2 Results of Implementation: Information

The project described in this thesis was, as stated in the ”Implementation” chapter, a cartography project, meaning that the main objective was the mapping of the existing architecture (separating design from representation). The main approach followed was the ”Enterprise Cartography” paradigm, described in [14] and [37]. Having a pre-defined, step by step, framework of how to conduct a EA project proved a to be a major enabler for the success of the project. Due to the size of the project, it would have been easy to become overwhelmed with the creation of a roadmap that defined what were the steps necessary and in what order.

While we can compare first hand how the project would have gone had we choose another framework, we can use time and smoothness (i.e not sudden halts in the overall progression of the project) as metrics to support the claim that this approach was adequate. Every step made logical sense for the EA team, as , for example, setting goals for the project before starting the project itself allowed the team to measure the success of the different phases, points of improvement, and to distribute resources accordingly.

Creating a base meta-model before starting the interviews with the teams allowed us to have some-

thing to show them, while their feedback also allowed us to create a meta-model that better represent their reality before beginning the mapping procedures.

After having concepts that allowed us to recreate reality, finding the best sources of information, and creating automatic methods/processes to extract information from them allowed the team to streamline the information collecting endeavours, since the main obstacle was defining what classes should exist and what relationships they had among them. These relationships were important to understand what views were relevant for the organization.

After having a structure that allowed for a more focused effort, we could them populate the repository with an initial baseline. Setting up mechanisms and concepts prior to the collection of information made the mapping process much easier. However, while this approach made tasks much faster, there was still the need to sometimes go back to a previous step of the framework, such as the creation of a new class in the meta-model, which the need only arose after the initial phase of the project.

Once the processes to obtain information were already well defined, the project then evolved in a iterative manner: the EA team would interview each team to inquire them about the structure of the system, what business processes it served, and what infrastructure supported it. These interviews were done personally for the main organization, and remotely to all the organizations belonging to the eSIS ecosystem. Once the information was imported into the ATLAS platform, it could them be visualized in the maps defined in the system.

4.2.1 Quantity of Information

Measuring the success of this part of the project was a hard task, since having collected huge amounts of information does not necessarily translate into good information (as in, information that in fact correctly describes reality and is helpful for those that use the Atlas platform).

As such, we decided to separate these two aspect to better rate the information collected. The first evaluation metric would be the quantity of the information. The goal set at the beginning of the project stated that ”at least, 25% of the systems within the organization should be mapped”. Due to the inconsistent scope of the project (new system’s were being developed each day, existing ones would be changed and some that were initially left out of the scope were afterwards included), this was a very vague objective. To simplify this evaluation, we state that the ”25%” refers to the initial expected number (as in, the number defined at the beginning of the project, without all the scope variations taken into account). By ”mapping systems” we mean that all 3 main views must be complete: ”Solution Integration”, ”Solution Context” and ”Solution Structure % Deploy”, and by complete we mean that all information required to built each of the maps exists within the Atlas platform. In this sense, we can state that the project was a success, as the initially defined 25 (out of approximately 100) systems were mapped. While one can argue that mapping 25 systems in 9 months times is an underwhelming number, it should be taken into account the whole already described preparatory process, such as the creation of the meta-model, development of the new features for atlas, teaching teams the concepts behind EA, automatizing the collection of information, testing these processes etc. As such, having a quarter of

the total systems (and their processes and infrastructure) mapped in (almost) their full detail can be considered a success.

To better understand the progression in terms of the amount of information collected, the following tables presents a comparison between the number of instances of each class at the beginning of the project (February 2018), and at the end of the author's participation (September 2018).

Class	Before	After	Comments
Actor	50	280	Mapping of the eSIS entities, as well as all the departments within the organization, along with profiles for the GDPR part. Some individuals also mapped due to their importance.
Business Process	27	40	To map processes, a macro approach was taken, meaning we did not map with the Processes with exhausting detail, hence the relative small number.
Business Functions	0	56	As explained in the GDPR solution implementation, one Business Function must exist for every Application function.
Application	430	1436	Mapping the systems was the main goal of this project, and with the help of the automatizing and streamlining of the collection of information, this class suffered the biggest jump in numbers.
Application Service	120	650	While the numbers suffered a positive increase, a lot of redundancy exists, as some services are not named after the generic interaction they represent (as defined in the meta-model), but the actual technical name and version.
Application Function	0	56	Used mainly to map GDPR concepts, such as the action one system can have over a certain type of information (Data object). Describes a very general type of action, hence the relative small number.
Solution Typology	0	12	Typologies are based on internal criteria defined by the Organization, and consist of set of 12 categories.
Data Object	25	480	Each Data object represents a piece of information used by the systems. These were described with detail, with some instances representing aggregators (ex: instance "Patient Personal Data" aggregates the instances "Biometric Data", "Blood type" etc.)

Class	Before	After	Comments
Data Typology	0	12	Classification based on internal criteria defined by the Organization. 12 Categories.
Environment	0	66	In practice this class was used to create maps within each of the repositories belonging to individual eSIS entities. As such, one for every entity was created
Node	105	385	Nodes created according to the descriptions of network engineers. More freedom was given to the members of the organization with this class since it did not have as many constraints, as for example, the application class had.
Node typology	0	12	Classification based on internal criteria defined by the Organization. 12 Categories.
Location	0	67	One instance for each physical location plus the ”cloud” location.
System Software	77	560	Much like the Application Service, the SS class contains a lot of redundancy, as users created instances that represent the same, but were misspelled or represent a slightly different version. However, the core instances, those that represent the generic software (ex: ”Microsoft Server 2003”) are more commonly use and satisfy the user needs in the views.
Project	0	24	Composed of projects that both represent the general aggregates and individual initiatives. based on the portfolio of projects the organization currently has in place (that the EA team had access to.)

4.2.2 Quality of information

The second part of this evaluation of the information regards how the information collected actually answers to the needs of the organization.

As previously discussed, information gathered and produced during the project was classified according to the number (quantity) and it's inherent capacity to provide the organization with the necessary information. While accurately measuring the ”quality” of the information is a difficult task, as there is no absolute metric to evaluate the information. To overcome this, we elaborated two characteristics to classify the quality of the information:

1. Conceptualization: Do meta-model concepts correctly represent the real-world concepts?
2. Accuracy: Do views created using the meta-model accurately represent the reality of the organization?

”Conceptualization” asks the users if the classes created for the meta-model can accurately map real world concepts, given their definition- For example, the vast majority of user agreed that the Application class accurately and consistently mapped concepts such as systems and parts of systems in a correct and consistent manner, meaning that for similar cases, the class was used to map those concepts. (Ex: Every system that had a front-end component, that component was mapped using the Application class. This was the case for every front-end part of any Solution within the eSIS. This means most of the users agreed with this decision and felt the Application class successfully mapped this concept). This characteristic also includes the relationships between the different classes, such as how a instance Application class that is hierarchically positioned in the middle (”Application”) is supported by a class of ”System Software”.

”Accuracy” refers to the way the maps created using the meta-model and the mappings performed by the EA team (with help from the organization) accurately represent the reality of it. Does a certain view, with its legends and concepts, correctly describes how the organization is structured.

While exact statistical data was hard to collect since forms had to be sent to every member of the organization, most of this classification was formulated based on the feedback provided to the EA team. The author set down with members of various (not all) teams and members f the organization’s EA team to understand their level of satisfaction with the information.

Members from the different departments and teams had a generally very positive opinion on the quality of the information, citing that there was still space for the meta-model to grow (”Conceptualization”) in order to incorporate every concept from the reality of the enterprise, but what existed already covered much of the concepts and allowed for an accurate and objective mapping of the reality. In regards to the maps, the team members were also generally very positive, describing the maps as very helpful tools to understand the relationship between different systems and infrastructure elements, with one of the common critics being the somewhat slow transition between maps (i.e when the user clicks on a class an wishes to a view where that class in the main subject, sometimes options were missing or the loading times between views were a longer than expected).

EA team members shared most of the opinions of department members, with an even more positive outlook on the project, citing that what was built up until the end of the author’s participation in the project showed ”immense potential”. The EA team also mentioned that conditions were met for the ”normalization of EA” to begin in a short period of time, meaning that the baselines for a sustainable and continuously improving EA were on their way to be met, meaning that the organization now felt conditions for turning the EA practice into a organic part of the organization were ”almost set”, mentioning that more information to populate the repositories was still necessary so that every repository was at the same level as the others. In terms of the existing views, the EA team members showed to be very satisfied with what existed, feeling it helped organization members, including themselves, to understand the reality of the organization, while mentioning that eventually more views would be necessary, as new requirements appeared.

4.2.3 Effectiveness of Solution

During the Implementation chapter (3), it was explained which measures were applied to tackle the issues/problems described in the Introductory chapter (1). In this section, a table will be provided with the solutions, and to what degree did help solve/mitigate the problems described at the beginning of the thesis. This evaluation is based on the previously mentioned organization feedback and the author's personal view on the matter.

The legends are as follows:

- 1 - Very Little Positive Influence
- 2 - Little Positive Influence
- 3 - Average Positive Influence
- 4 - Considerable Positive Influence
- 5 - Very Positive Influence

Solution/Problem	1.2.1	1.2.2	1.2.3	1.2.4	1.2.5	1.2.6	1.2.7
3.1	2	3			5		
3.2	2	5					
3.3		2		5			
3.4			5			4	
3.5			5			4	
3.6	2		5				
3.7					3	5	
3.8							5

The effect each solution had on the respective problems can be summarized in the following way (for a more in-depth description, please see the respective sub-chapter for each solution in Chapter 3 "Implementation"):

3.1 - Standardization of Concepts/Language

The creation of a meta-model to be used within the organization facilitated the rest of the EA effort, since it is one of the most demanding parts of the project, which took weight off the shoulders of the organization members, providing them with additional motivation (Problem 1.2.1 - "SPMS and Ecosystem members show lack of motivation to perform EA effort").

The process of collecting information also requires the organization of said information. By already having a defined meta-model, classifying and mapping collected information becomes easier, as we simply have to map the information to the most appropriate class (Problem 1.2.2 - "Information collecting for EA reveals to be a cumbersome task").

And of course, defining a meta-model helps create a single uniform language to be used by all of the organization (Problem 1.2.5 - ”No uniform language to define concepts of the organizations reality”).

3.2 - Utilization of Forms to Streamline Information Collecting

The creation of a much simpler and easier way to input information into the Atlas platform’s repositories, which came in the form of Forms, helped streamline the effort needed from the members of the organization, by making it less technical and preventing user mistakes (to a certain degree) (Problem 1.2.1 - ”SPMS and Ecosystem members show lack of motivation to perform EA effort”).

This new way of inputting information also made the process of collecting information much more simpler than before, successfully making this task much more effortless (Problem 1.2.2 - ”Information collecting for EA reveals to be a cumbersome task”)

3.3 - Separation of Design and Representation

The separation of design and representation allowed the team to focus more on representation (how the information is presented to the users) than design. As such, the focus was on creating and maintaining the maps. This was successfully achieved by configuration maps using the Atlas tool, which functions on query basis, meaning that maps are automatically created, using the queries written by the EA team to group information that respects the constraints of information imposed by the query. This means that maps are dynamic, and so, if new information that fits the description of the information the map must present, this new data will automatically be added to the view, ensuring it does not require constant redrawing (Problem 1.2.4 - ”Views generated by EA become obsolete very fast”).

3.4 - Use of Atlas to Support Management

The introduction of mechanisms in the Atlas tool focused on supporting the management practice, such as the Charts, or even architectural maps to study dependencies, relation between systems, allowed for the organization to solve management operational problems, meaning that for someone whose responsibility is management, Atlas tools presents information in such way that it directly helps in their daily work (Problem 1.2.3 - ”SPMS struggles to find way to use EA to solve operational problems”) .

The information presented in these charts (which can present KPIs) and maps helps the create EA awareness and provide information relevant for the management of a EA project, since the Atlas platform presents this information in an immediate, holistic or detailed way, creating a set of controls that allow for the monitoring of the EA effort and help better understand the stakes and concepts that influence and should be taken into account in such effort (Problem 1.2.6 - ”Lack of awareness of what the main point regarding the management of an EA effort should be”).

3.5 - Use Atlas to Support Planning

Much like the previous solution, Atlas now presents information that directly helps solve/support operational problems for those whose purpose is to create plans for the future. Information presented in maps,

the option to attribute a set of dates to each instance allows the user to have a view of the past, present and future (Problem 1.2.3 - ”SPMS struggles to find way to use EA to solve operational problems”).

The presentation of information regarding planning in a structured, direct and simple way also helped the organization better understand the purpose of the EA effort and how planning this effort could be done, therefore making this process much more clear (Problem 1.2.6 - ”Lack of awareness of what the main point regarding the management of an EA effort should be”).

3.6 - Use EA to Solve Operational Problems

In this project, a way to directly use EA to solve operational necessities besides those of management were also implemented. They came in the forms of the introduction of new meta-model concepts in order to make the mapping of GDPR concepts possible. GDPR is an obligatory regulation the organization must abide to. While the views and instances created within the Atlas platform do not suffice to prove compliance to the regulation, it provides an overview of how the organization organizes accesses and profiles, something that must be shown to the authorities in the case of an audit. Therefore, mapping these concepts and their relations is an operational necessity (Problem 1.2.3 - ”SPMS struggles to find way to use EA to solve operational problems”).

The fact that Atlas also supports mapping GDPR concepts proved the utility and flexibility of the tool, which in turn made members of the organization feel more secure about the EA project, motivating them to help the project, as they felt there were benefits in it for them too. (Problem 1.2.1 - ”SPMS and Ecosystem members show lack of motivation to perform EA effort”).

3.7 - Document Relevant Points of the Project

Throughout the project, relevant documentation has been created about the effort, such as user manuals, weekly reports, technical guides etc. This documentation helped to communicate the meta-model through all of the organization and Ecosystem, since it explained in great detail the concepts behind it, the purpose of them, while also explaining mechanisms such as the views and charts. (Problem 1.2.5 - ”No uniform language to define concepts of the organization’s reality ”).

The documentation also helps the organization and entities gain more awareness of the EA practice, since it documents the ups and downs of the project, the steps taken towards the solution, and help create a global vision of the purpose of the project, and how it should be performed. (Problem 1.2.6 - Lack of awareness of what the main point regarding the management of an EA effort should be”).

3.8 - Use of Charts, Information Structuring Rules and Pilot Testing to Facilitate Multi-repository

This solution was specifically design to tackle the problems that come with managing a multi-repository project, specially in a scenario with many repositories. The implementation of automatic upload of information features, quality-control tools such as the 3 step upload process (see chapter 3.8.3), the analysis of meta-data to guarantee information consistency (with the Charts feature), as well as the implementation of a scenario manager, that allowed repository managers to preview change proposals

without direct effect on the information that already exists in the repository, and the testing of such features with the help of pilot testing, contributed in a very positive way for the management of a multi-repository reality and the successful tailoring of theses solutions for the specific challenges (Problem 1.2.7- ”Difficulty coordinating multiple repositories”).

Chapter 5

Conclusions

In this thesis we explored the specific problems that an organization struggled with for not having an EA in place and properly managed . To solve/mitigate these issues, we created a set of solutions based on the framework proposed in [14, 37]. The results, while far from definitive, since the full implementation of an EA into an organization (when it becomes an important part of the organization , meaning it becomes something that is used consistently and managed and maintained in that same way) is a process that can take a very long time, since this effort varies with the complexity of the organization, as well as it's size.

The author came to the conclusion that the implementation of an EA is a effort that requires not only a technical aspect, such as knowing what tools to use, ways to collect information more easily or building the meta-model, but also a ”political” effort. People within organization have different ideas and agendas, which translate into opinions that are sometime hard to balance, which in our case showed that in some cases, one must resort to pressuring the upper hierarchy of the organization to make sure progress is made and that changes are accepted.

While it can't be said with all the certainty that the framework utilized to perform this project was the most efficient(since no other was used to provide comparative data), we can say that it proved to be a very efficient and structured one, providing clear guidelines for those in responsibility to follow. While the framework effectiveness depends on how well the person using it performs each step, it's main point seemed to be the clear structuring of which step to take in which order, and how each previous step facilitates the performance of the next one. The fundamental of the framework, the separation between design and representation, proved to be an effective method to tackle these projects, since focusing only on the representation (cartography) allowed for a much more focused effort, since the design, such as the logic of how best to organize systems, actors, infrastructure etc. was not a concern in this project.

Another relevant point was the fact that the author successfully confirmed (for this specific project) the accuracy of the description of a EA Project verified in [9], experiencing first hand that EA projects, are indeed, different from traditional IT projects in their various characteristics such as Scope (very loosely defined), Change control (many stakeholders, many change requests, difficult to keep track of these) and Speed changes (very high). The author also came into contact with described failure factors [10]

that ”help” EA projects fail, such as the lack of EA awareness, limited commitment from some of the interested parties and the time it takes to set up (in this case map) the EA.

We also come to the conclusion that using an enterprise architecture tool is absolutely fundamental, as a majority of the solutions were only possible due to the existence of the Atlas tool in the project. The tool also proved itself of being key to the success by providing a very flexible structure, meaning changes and updates to the software were relatively easy to implement, which allowed to the tool to better fit the needs of the target organization. Mechanisms to automatically import and upload information from one repository to another, quality control, personalized views and management & Planning tools proved essential to the success of the project.

The coordination of a large number of repositories also proved to be the biggest challenge in this project. While mechanisms to control and upload the information are in place, these can only be upgraded by experience, meaning that one must face the headaches that will certainly come with the growth of the information and the need to maintain consistency between repositories. The solutions implemented to tackle this specific issue are based on small pilot tests and previous experience with similar situations. However, there is no guarantee that they will suffice for the rest of the duration of the EA initiative, and as such will require monitoring by those responsible for the system, and in the future, the implementation of more effective control techniques.

5.1 Achievements

Upon finishing my participation in the project, the following major milestones were accomplished:

- Creation of meta-model that is currently in use by all of the eSIS ecosystem entities (along with its documentation). This meta-model allowed for communication to standardized and concepts mapped.
- Creation of 20+ views/maps, tailored to the necessities of members of the organization. These provide an important strategic and management tool, as they are no static and can change dynamically through management approved changes.
- Successful implementation of Forms, a simpler and easier way for users to register information in the Atlas platform.
- Identified and capitalised on strategic sources of information (existing excel files, VIPs in each entity) to facilitate the upload of information into the Atlas repositories.
- Implementation of systems within the Atlas tool that guarantee information quality and consistency, along with documentation on these.
- Introduction of EA concepts and mindset in an organization with no previous experience, while introducing features that allow EA to play a role more operational tasks.

- Fulfilled the objective of having 25% of the initially agreed systems accurately mapped in the Atlas tool.
- Kick-started an EA initiative that covers over +66 public health-care entities.
- Created comfortable baselines for the rest of the project (meta-model, upload mechanisms, quality control, views etc).
- Created entire multi-repository structure, capable of centralised information and quality control features, guaranteeing a valuable network for information gathering and management.

5.2 Future Work

In the future, one can consider topics such as how the effectiveness of the framework used in this project directly compares to others, and for example, which topics of the project change, such as the scope, the change control, requirements, stakeholders etc.

Bibliography

[1] R. L. Daft. *Organic versus Mechanistic Management Processes*. 2010. ISBN 9780324598896. doi: 10.1017/CBO9781107415324.004. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0.

[2] Judith Jones. Enterprise Architecture Maturity Models. 2003. URL http://www.opengroup.org/architecture/0310wash/presents/Judith_Jones-Maturity_Models.pdf.

[3] The Open Group. 51. Architecture Maturity Models, 2011. URL http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap51.html.

[4] Serviço Nacional de Saúde. Entratégia Nacional para o Ecossistema de Informação de Saúde (http://enesis.spms.min-saude.pt/wp-content/uploads/2017/07/brochura-online_v1.pdf), 2017. URL http://enesis.spms.min-saude.pt/wp-content/uploads/2017/07/brochura-online_v1.pdf.

[5] S. Gregor, D. Hart, and N. Martin. Enterprise architectures: enablers of business strategy and IS/IT alignment in government. *Information Technology & People*, 20(2):96–120, 2007.

[6] E. Niemi. Enterprise architecture benefits: Perceptions from literature and practice. *7th IBIMA Conference Internet & Information Systems in the Digital Age*, 14:16, 2006.

[7] V. Boucharas, M. Van Steenbergen, S. Jansen, and S. Brinkkemper. The contribution of enterprise architecture to the achievement of organizational goals: A review of the evidence. *Lecture Notes in Business Information Processing*, 70 LNBIP(June):1–15, 2010. ISSN 18651348. doi: 10.1007/978-3-642-16819-2{_}1.

[8] J. M. Morganwalp and A. P. Sage. Enterprise architecture measures of effectiveness. *International Journal of Technology, Policy and Management*, 4, 2004.

[9] F. Anajafi, R. Nassiri, and G. Shabgahi. Developing effective project management for enterprise architecture projects. 1:1–388, 2010. ISSN 9781424486663. doi: 10.1109/ICSTE.2010.5608867. URL http://ieeexplore.ieee.org/ielx5/5598555/5608837/05608867.pdf?tp=&arnumber=5608867&isnumber=5608837%5Cnhttp://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5608867&newsearch=true&queryText=.QT.enterprisearchitecture.QT.and.QT.projectmanagement.Q.

[10] S. Roeleven. Why Two Thirds of Enterprise Architecture Projects Fail An explanation for the limited success of architecture projects. (December):12, 2010.

[11] Christy Pettey and Rob van der Meulen. Gartner Identifies Ten Enterprise Architecture Pitfalls, 2009. URL https://www.gartner.com/newsroom/id/1159617.

[12] Ivar Jacobson. Enterprise Architecture has failed in a big way - TechRepublic, 2007. URL https://www.techrepublic.com/article/enterprise-architecture-has-failed-in-a-big-way/.

[13] W. A. Günter. Measuring Enterprise Architecture Effectiveness: A Focus on Key Performance Indicators. page 220, 2014. URL http://liacs.leidenuniv.nl/assets/Masterscripties/ICTiB/Gunther-Wendy-non-confidential.pdf.

[14] P. Sousa and J. Tribolet. Enterprise Cartography : From Theory to Practice. 2014.

[15] E. P. Martin Op't Land. *Enterprise Architecture - Creating Value by Informed Goveranance*. 2009. ISBN 9783540852315. doi: 10.1007/978-3-540-85232-2.

[16] Michael Schrage. Are You Driving Too Much Change, Too Fast?, 2012. URL https://hbr.org/2012/11/are-you-driving-too-much-change.html.

[17] T. Hope. The Critical Success Factors of Enterprise Architecture. *University of Technology, Sydney*, pages 1–396, 2015.

[18] Federation of Enterprise Architecture Professional Organizations. A Common Perspective on Enterprise Architecture. *Architecture and Governance Magazine*, 9:6, 2013. ISSN 1387-3326. doi: 10.4018/978-1-60566-659-4.

[19] M. Lankhorst. *Enterprise Architecture at Work - Enterprise Modelling, Communication and Analysis - Second Edition*, volume 36. 2013. ISBN 9783642296505. doi: 10.1016/B978-0-12-387667-6.00013-0. URL http://www.ncbi.nlm.nih.gov/pubmed/24815723.

[20] Zak Cole. The Economic Value of Enterprise Architecture and How to Show It — Erwin (https://erwin.com/blog/economic-value-of-enterprise-architecture/), 2016. URL https://erwin.com/blog/economic-value-of-enterprise-architecture/http://erwin.com/blog/economic-value-enterprise-architecture-show/.

[21] J. C. Henderson and H. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1):472–484, 1993. ISSN 0018-8670. doi: 10.1147/sj.382.0472. URL http://ieeexplore.ieee.org/document/5387398/.

[22] A. I Bleeker, H. Proper, and S. Hoppenbrouwers. The Role of Concept Management in System Development – A practical and a theoretical perspective, 2004.

[23] M. Lankhorst. State of the Art. *Enterprise Architecture at Work*, pages 61–74, 2013. ISSN 14712407. doi: 10.1007/978-3-642-29651-2. URL http://link.springer.com/10.1007/978-3-642-29651-2.

[24] M. Janssen and B. Klievink. Can enterprise architectures reduce failure in development projects? *Transforming Government: People, Process and Policy*, 6(1):27–40, 2012. ISSN 1750-6166. doi: 10.1108/17506161211214804. URL http://www.emeraldinsight.com/doi/10.1108/17506161211214804.

[25] B. Burton, Marcus Blosch, Bryan Burke, and Lee Weldon. Leadership Vision for 2018: Enterprise Architecture and Technology Innovation Leader. 2017. URL https://www.gartner.com/doc/3623317/leadership-vision-enterprise-architecture-technology.

[26] S. W. Ambler and L. L. Constantine. *The Unified Process Elaboration Phase: Best Practices in Implementing the UP*. Best practices in implementing the UP. Taylor & Francis, 2000. ISBN 9781929629053. URL https://books.google.pt/books?id=f4q0rkM44XAC.

[27] Techniques and Methodologies Group. UN/CEFACT's Modeling Methodology (UMM): UMM Meta Model – Foundation Module Version 1.0 Technical Specification 2006-10-06. *Meta*, 2006.

[28] B. Hofreiter, C. Huemer, and K.-D. Naujok. UN/CEFACT's Business Collaboration Framework - Motivation and Basic Concepts. *Coordination and Agent Technology in Value Networks. Tagungsband zur Multi-Konferenz Wirtschaftsinformatik (MKWI 04)*, pages 93–108, 2004.

[29] T. O. Group. TOGAF Version 9.1. Online, 2011. URL http://pubs.opengroup.org/architecture/togaf9-doc/arch/.

[30] J. A. Zachman. The Concise Definition of The Zachman Framework by: John A. Zachman (https://www.zachman.com/16-zachman/the-zachman-framework/35-the-concise-definition), 2008. URL https://www.zachman.com/16-zachman/the-zachman-framework/35-the-concise-definition.

[31] ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems. *ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15*, pages 1–24, 2007. doi: 10.1109/IEEESTD.2007.386501.

[32] Sparx. Enterprise Architect - UML Design Tools and UML CASE Tools for Software Development, 2016. URL http://sparxsystems.com/products/ea/http://www.sparxsystems.com/products/ea/.

[33] IBM. IBM Knowledge Center - Overview of IBM Rational System Architect (https://www.ibm.com/support/knowledgecenter/en/SS6RBX_11.4.3/com.ibm.sa_base.legal.doc/topics/rsysarch_ove 2018. URL https://www.ibm.com/support/knowledgecenter/en/SS6RBX_11.4.3/com.ibm.sa_base.legal.doc/topics/rsysarch_overview_base.html.

[34] Plainview. Enterprise Architecture Software — EA Tools — Watch Demo — Planview (https://www.planview.com/tapm-technology-application-portfolio-management/), 2015. URL https://www.planview.com/tapm-technology-application-portfolio-management/.

[35] Julie Short and Chris Wilson. Enterprise Architecture Tools Are Positioned to Deliver Business Value. Technical report, 2011. URL https://www.gartner.com/doc/1728043/enterprise-architecture-tools-positioned-deliver.

[36] P. Johnson, E. Johansson, T. Sommestad, and J. Ullberg. A Tool for Enterprise Architecture Analysis. *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pages 142–142, 2007. ISSN 1541-7719. doi: 10.1109/EDOC.2007.25. URL http://ieeexplore.ieee.org/document/4383989/.

[37] J. Tribolet, P. Sousa, and A. Caetano. The Role of Enterprise Governance and Cartography in Enterprise Engineering. *Emisa*, 0(0):1–13, 2014.

[38] R. Stichweh. Systems Theory. pages 1–14, 2010. doi: 10.4135/9781446270141.n1.

[39] S. Buckl, A. M. Ernst, F. Matthes, and C. M. Schweda. How to make your enterprise architecture management endeavor fail! *Proceedings of the 16th Conference on Pattern Languages of Programs - PLoP '09*, page 1, 2010. doi: 10.1145/1943226.1943229. URL http://portal.acm.org/citation.cfm?doid=1943226.1943229.

[40] D. H. Olsen and K. Trelsgård. Enterprise architecture adoption challenges: An exploratory case study of the norwegian higher education sector. 2016.